TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Aplikace pro zpracování měřených dat pro mobilní platformu Android

Diplomová práce

Studijní program: N2612 – Elektrotechnika a informatika
Studijní obor: 1802T007 – Informační technologie

Autor práce: **Bc. Adam Smolík**
Vedoucí práce: Ing. Jan Kraus, Ph.D.

Liberec 2016

TECHNICAL UNIVERSITY OF LIBEREC
Faculty of Mechatronics, Informatics
and Interdisciplinary Studies

Application for Measured Data Management for Android Mobile Platform

Diploma thesis

Study programme: N2612 – Electrical Engineering and Informatics
Study branch: 1802T007 – Information Technology

Author: **Bc. Adam Smolík**
Supervisor: Ing. Jan Kraus, Ph.D.

Liberec 2016

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky a mezioborových studií
Akademický rok: **2015/2016**

ZADÁNÍ DIPLOMOVÉ PRÁCE
(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Bc. Adam Smolík**
Osobní číslo:	**M12000225**
Studijní program:	**N2612 Elektrotechnika a informatika**
Studijní obor:	**Informační technologie**
Název tématu:	**Aplikace pro zpracování měřených dat pro mobilní platformu Android**
Zadávající katedra:	**Ústav mechatroniky a technické informatiky**

Zásady pro vypracování:

1. Seznamte se s požadavky na odečet a archivaci dat z paměti analyzátorů kvality.
2. Seznamte se s problematikou návrhu aplikací pro zvolená mobilní zařízení s OS Android, vyberte vhodné nástroje pro vývoj vlastní GUI aplikace pro záznam a zobrazení hodnot měření analyzátorů platformy F4 v2.0.
3. Navrhněte a vytvořte aplikaci pro odečet, záznam, přenos a základní vyhodnocení aktuálních a archivních dat z výše uvedených analyzátorů.
4. Diskutujte výhody či nevýhody vytvořeného díla. Uveďte konkrétní možnosti dalšího rozvoje implementovaných metod a možnosti či omezení pro jejich využití v praxi.

Rozsah grafických prací: **dle potřeby dokumentace**

Rozsah pracovní zprávy: **cca 40–50 stran**

Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam odborné literatury:

[1] **KRAUS, Jan a Martin BLÍŽKOVSKÝ. KMB SYSTEMS, S.R.O. Uživatelská příručka aplikace ENVIS v. 1.2 [online]. 2015. [cit. 2015-10-08]. Dostupné z: http://www.kmb.cz/**

[2] **BLOCH, Joshua. Effective Java. 2nd ed. Upper Saddle River, N.J.: Addison-Wesley, c2008, xxi, 346 s. Java Series. ISBN 978-0-321-35668-0.**

[3] **GOETZ, Brian. Java concurrency in practice. Upper Saddle River, NJ: Addison-Wesley, c2006, xx, 403 p. ISBN 9780321349606.**

[4] **FREEMAN, Eric, Elisabeth FREEMAN, Kathy SIERRA a Bert BATES. Head first design patterns. Sebastopol, CA: O'Reilly, c2004, xxxvi, 638 p. ISBN 0596007124.**

Vedoucí diplomové práce: **Ing. Jan Kraus, Ph.D.**
Ústav mechatroniky a technické informatiky

Datum zadání diplomové práce: **10. října 2015**

Termín odevzdání diplomové práce: **16. května 2016**

prof. Ing. Václav Kopecký, CSc.
děkan

L.S.

doc. Ing. Milan Kolář, CSc.
vedoucí ústavu

V Liberci dne 10. října 2015

Prohlášení

Byl jsem seznámen s tím, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé diplomové práce pro vnitřní potřebu TUL.

Užiji-li diplomovou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Diplomovou práci jsem vypracoval samostatně s použitím uvedené literatury a na základě konzultací s vedoucím mé diplomové práce a konzultantem.

Současně čestně prohlašuji, že tištěná verze práce se shoduje s elektronickou verzí, vloženou do IS STAG.

Datum: 15.5.2016

Podpis: Gnolik

Abstrakt

Práce se zabývá vývojem mobilní aplikace pro systém Android, která bude komunikovat s analyzátory platformy F4 v2.0 od firmy KMB systems. V úvodu práce byly zjištěny možnosti komunikace s analyzátorem a struktura ukládaných souborů. Byly rozebrány návrhové vzory vhodné k vývoji aplikací pro Android a popsány základní stavební komponenty k tvorbě mobilních aplikací. Ze tří popisovaných návrhových vzorů MVC, MVP a MVVM byl zvolen MVVM. Samotná aplikace byla rozdělena na dvě části: knihovnu, umožňující komunikaci a záznam dat, a Android aplikaci implementující zmíněnou knihovnu. Mezi hlavní funkce aplikace patří komunikace s analyzátorem a odečet dat. Data jsou ukládána do přenositelných souborů. Dále aplikace implementuje jednoduchou databázi, která poskytuje uživateli snadnou správu jeho analyzátorů. Aplikace také poskytuje přehled o aktuálně měřených datech, která zobrazuje do tabulek. Aktuální data jsou pravidelně obnovována podle uživatelsky volitelné frekvence. Výsledkem práce je mobilní aplikace umožňující záznam a přenos archivů a zobrazení aktuálních dat.

Klíčová slova

Záznam dat analyzátoru; analyzátor kvality energie; Android aplikace; aktuální data; návrhové vzory

Abstract

This work and document deals with the development of mobile application for Android. The application will communicate with the analyzers of the platform F4 v2.0 from company named the KMB systems. First, the possibilities of communication and the structure of file were examined. Next step was the analysis of software design patters for Android mobile applications. The application was divided into two parts. First part is library which focus on the communication and creation of archives. Second part was the development of mobile application which implements this library. The main function of application is communication with analyzer and the downloading of records. The records are saved into transferable files. Application implements a simple database. The database allows user to manage his devices. User can also see the actual data. Actual data are displayed in tabs and these tabs are refreshed at regular intervals. User can choose the refresh interval. The result of work is application which allows the download and transfer of records.

Keywords

Recording data analyzer; power quality analyzer; Android application; actual data; software design patterns

Obsah

Seznam obrázků

Seznam ukázek kódu

Seznam zkratek

- CEA (Compressed Envis Archive) – soubor obsahující data z analyzátoru
- API (Application Programming Interface) – rozhraní pro programování aplikací
- MVC (Model View Controller) – návrhový vzor
- MVP (Model View Presenter) – návrhový vzor
- MVVM (Model View ViewModel) – návrhový vzor
- XML (eXtensibile Markup Languagu) – obecný značkovací jazyk
- WI-FI (Wireles Fidelity) – bezdrátová komunikace v počítačových sítích
- AP (Access point) – přístupový bod do sítě
- DAO (Data Access Object) – mechanismus přístupu do databáze
- USB (Universal Serial Bus) – moderní způsob připojení periferií k počítači
- RAM (Random Access Memory) – polovodičové paměti s přímým přístupem
- URL (Uniform Resource Locator) – specifikace umístění zdrojů informací na internetu
- CRC (Cyclic Redudancy Check) – hašovací funkce
- GPS (Global Positioning System) – systém k určení geografické polohy přijímače
- UI (User Interface) – uživatelské rozhraní
- BCD (Binary Coded Decimal) – dvojkově reprezentované dekadické číslo
- UTD (Universal Type Description) – univerzální struktura popisující proměnnou

1 Úvod

Žijeme v době moderních technologií, které nás obklopují na každém našem kroku. Nejvíce progresivním vynálezem posledních let je mobilní telefon. Svoji cestu si našel i v oblasti průmyslu. Tablety a mobily díky vysokému výkonu a skvělým zobrazovacím schopnostem mohou sloužit jako vzdálené displeje. Využití jistě nalezne i množství senzorů jako jsou GPS, akcelerometr, gyroskop, senzor přiblížení, senzor světelnosti a magnetický senzor.

Cílem této práce je vytvořit mobilní aplikaci pro systém Android, která bude komunikovat s analyzátory platformy F4 v2.0 od firmy KMB Systems. Analyzátory ukládají měřené hodnoty, ale samy o sobě nemají velké zobrazovací schopnosti. Tento projekt se zaměřuje na zjednodušení práce s analyzátory, odečet měřených dat, jejich přenos a základní vyhodnocení aktuálních a archivních dat z výše uvedených analyzátorů.

Na začátku práce bude rozebráno zadání a z něho vytvořen seznam důležitých funkcí, které by měla aplikace implementovat. Mezi tyto funkce patří správa uživatelských zařízení, připojení k zařízení a zobrazení aktuálního stavu, stažení zaznamenaných dat, vytvoření validního CEA souboru a zobrazení aktuálně měřených dat. Pro potřeby aplikace bude navržena knihovna určená pro komunikaci s přístrojem a tvorbu CEA souborů. Dále budou ukázána a porovnána řešení problematiky od jiných firem. Na závěr budou provedeny testy aplikace na různých zařízeních a jejich vyhodnocení.

Samotná mobilní aplikace bude navrhována pro operační systém Android, proto budou ukázány a diskutovány nejčastěji používané návrhové vzory. Konkrétně budou probírány Model View Controller, Model View Presenter a Model View ViewModel. Práce se také zaměří na Android, zejména

na nejčastěji používané komponenty. Během vývoje aplikace byly použity také knihovny třetích stran pro ulehčení práce s návrhovým vzorem, s databází a se stavy aplikace. Použité knihovny jsou Android-ViewModelBinding, Android-StatefulLayout, ActiveAndroid a BindingCollectionAdapter.

2 Rozbor zadání, požadavky na aplikaci a existující řešení

Zadání jasně definuje, co je cílem této diplomové práce. Nejprve je třeba se seznámit s požadavky na odečet a archivaci dat z paměti analyzátorů kvality. Dále je potřeba se seznámit s problematikou návrhu aplikací pro zařízení s operačním systémem Android a vybrat vhodné nástroje pro vývoj vlastní GUI aplikace pro záznam a zobrazení hodnot měření analyzátorů platformy F4 v2.0. Následně realizovat aplikaci pro odečet, záznam, přenos a základní vyhodnocení aktuálních a archivních dat z výše uvedených analyzátorů. Nakonec diskutovat výhody a nevýhody vytvořeného díla a uvést konkrétní možnosti dalšího vývoje.

2.1 Požadavky na aplikaci

Dle zadání a jeho rozboru bylo nutné vytvořit komunikační knihovnu v jazyce Java. Komunikační knihovna zajišťuje snadné stažení dat z analyzátoru pomocí protokolu KMBLong, jejich vyhodnocení a vytvoření přenositelného CEA souboru. Komunikace s přístrojem musí být navržena tak, aby zvládla přenos většího množství dat. Tuto knihovnu bude implementovat mobilní aplikace navržená pro operační systém Android. Aplikace bude uchovávat informace o analyzátorech tak, aby nebylo nutné pokaždé zadávat informace potřebné pro připojení. Bude umožňovat jednoduchou uživatelskou správu zařízení.

Aby bylo možné uchovávat zařízení, bude nutné navrhnout jednoduchou databázi. Uživatel ji bude používat k přidávání, odebírání a editaci svých zařízení. Aplikace bude obsahovat základní informace v offline módu. Pro podrobnější stav zápisu dat, obsazení paměti přístroje a zobrazení aktuálních záznamů bude nutné se připojit k přístroji a požadovaná data

stáhnout. K vytvoření CEA souboru bude implementována služba, která dokáže běžet nezávisle na životním cyklu aplikace a data stáhnout na pozadí. Zobrazení aktuálních dat bude probíhat pomocí opakovaného dotazu podle uživatelem určené frekvence.

2.2 Podobná nebo existující řešení

V současnosti existuje mnoho řešení tohoto problému. Většina těchto řešení využívá analyzátoru zapojeného do elektrického obvodu, který je přes Ethernet připojen k internetu. Některá zařízení dokonce implementují vlastní Wi-Fi AP. K analyzátoru se nejčastěji připojuje pomocí desktopové nebo mobilní aplikace.

Firma Schneider Electric's nabízí zdarma aplikaci na iTunes. Ta by měla uživateli usnadnit pochopení elektrické spotřeby pomocí infografiky. Slibuje snadné použití pro rodiny, které by díky ní měly zjistit, proč mají vysokou spotřebu. Aplikace je nicméně nabízena pouze pro operační systém iOS a působí zastaralým dojmem. [1]

Řešení od společnosti Wattwatchers slibuje přesné měření obnovované každých pět sekund. Umožňuje měřit energii, spotřebu, frekvenci, napětí atd. Zařízení je připojeno do sítě přes Wi-Fi, 3G nebo Ethernet a lze s ním komunikovat pomocí desktopové aplikace nebo mobilní iOS aplikace. GUI aplikace působí zastarale a nepříliš uživatelsky přívětivě. [2]

S velmi moderním řešením přišla firma Curb. Pomocí analyzéru připojeného k internetu a elektrické síti umožňuje nejen sledovat a vyhodnocovat aktuální data, ale dokáže i odesílat notifikace do zařízení, pokud je spotřeba vyšší než obvykle. Aplikace vypadají velice moderně, jsou navrženy především pro produkty firmy Apple, včetně Apple Watch. [3]

Kompletní řešení nabízí firma Dranetz, jejíž analyzátory se vyznačují velkým dotykovým displejem, který slibuje podobné ovládání jako moderní tablety. Analyzátory se nastavují automaticky a k internetu je lze připojit přes Ethernet, USB a volitelně i přes Bluetooth. Přístroje společnosti Dranetz uživateli umožňují vzdáleně sledovat data a měnit nastavení pomocí chytrého telefonu, tabletu, PC nebo MACu. Podporovány jsou mobilní přístroje s operačním systémem Android i iOS. [4]

Jednoduchý záznam dat nabízí firma Kyoritsu. Její analyzátory disponují menším barevným displejem, který umožňuje zobrazení aktuálních dat, a ovládacími tlačítky. Zajímavostí je tlačítko print green, které uloží obsah displeje jako obrázek. Zařízení umožňuje záznam dat na externí SD kartu. Ta může být použita pro následný přenos dat. Mimo jiné analyzátor umožňuje připojení a přístup k datům přes bluetooth a USB pro mobilní zařízení se systémem Android a PC. [5]

Firma HT nabízí inovativní přenosné přístroje, které dovedou snadno analyzovat třífázové a jednofázové elektrické sítě. Přenosné zařízení nevyžaduje žádné nastavování, stačí ho připojit a okamžitě začne měřit. Má vlastní zdroj napájení ve formě baterie, která se automaticky dobíjí během měření. Analyzátor se chlubí IP65 ochranou, která mu umožňuje pracovat v jakémkoli prostředí. Zaznamenaná data lze přenést pomocí Wi-Fi připojení na mobilní telefon, tablet či PC. Alternativa k tomuto přenosu je USB konektor, který umožňuje připojení PC. Přenos dat pro mobilní telefony a tablety je zajištěn pomocí Android a iOS aplikace. [6]

Malé přenosné měřiče od firmy PowerSight využívají iPad jako zobrazovací zařízení. Pomocí aplikace se lze připojit k měřícímu přístroji přes Wi-Fi a vyhodnotit veškerá data z bezpečné vzdálenosti od měřeného zařízení.

Tablet zobrazuje data pomocí generovaných tabulek a grafů. Kromě tabletu se lze k analyzátoru připojit přes PC s Wi-Fi. [7]

3 Teoretická část

V teoretické části bude popsána struktura CEA souboru, rozebrán komunikační protokol KMBLong, porovnány návrhové vzory pro vývoj Android aplikace a krátce popsány komponenty použité při vývoji aplikace.

3.1 Analyzátor SMC 144

Přístroj SMC 144 je navržen pro dálkové sledování a záznam spotřeby elektrické energie, její řízení a monitorování kvality. Do vnitřní paměti ukládá zátěžové profily, odečty elektroměru, události a pravidelně i všechny ostatní měřené veličiny. Přenos dat a nastavení přístroje lze provádět přes USB, 2x RS-485 a Ethernet. Data lze zobrazit v programu ENVIS.

3.2 KMBLong komunikační protokol

Komunikační kanál používá osm datových bitů, bez parity a s jedním stop bitem. Adresa a datový tok mohou být nastaveny. Komunikační protokol využívá master-slave filozofii. Analyzátor odesílá relevantní odpověď po přijetí zprávy. Veškeré podporované zprávy mají jednotný formát: adresa zařízení (1 bajt), délka zprávy (2 bajty), typ zprávy (1 bajt), tělo zprávy v závislosti na typu zprávy a 16-ti bitový kontrolní součet CRC. Pokud nastal problém, je typ zprávy v odpovědi nastaven na nulu a tělo zprávy obsahuje jeden bajt s kódem chyby. V případě, že nenastal problém, analyzátor odesílá odpověď podle typu přijaté zprávy. Veškeré hodnoty jsou posílány jako Big Endian.

Jedním z typů zprávy je identifikace zařízení. Po jejím přijetí odpoví analyzátor zprávou obsahující základní informace o zařízení. Používá se pro výčet zařízení, aktualizaci firmwaru, výměnu dat a předchází většinu běžných úkolů.

Zpráva pro získání aktuálních dat může obsahovat konfigurační masku. Tato maska definuje strukturu dat, která budou odeslána v těle odpovědi.

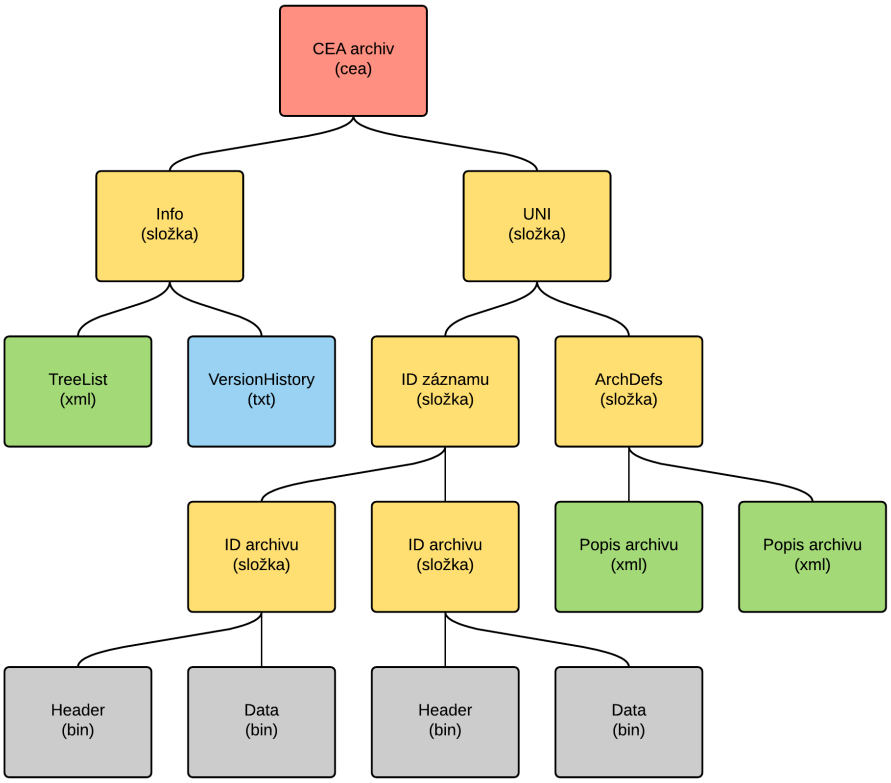
Všechna měřená data lze rozložit do několika požadavků k zlepšení výkonu. Odpověď obsahuje masku požadavku a zvolená data.

Zprávy pro získání konfigurace se dělí na SmpInstallConfig a SmpConfig. Struktura SmpInstallConfig obsahuje základní instalační hodnoty, které definují transformační poměry a nominální hodnoty pro napětí a výkon. SmpConfig obsahuje dodatečné nastavení jako nastavení komunikace, vlastnosti displeje, nastavení časové zóny, přístupy pro administrátory a konfiguraci průměrování měření. [8]

3.3 CEA soubor

CEA soubor je soubor reprezentující výstup dat z analyzátoru. Můžeme ho vytvořit pomocí knihovny, kterou popisuje tato práce, nebo pomocí programu ENVIS od společnosti KMB systems.

Data získaná z analyzátoru jsou ukládána do CEA souboru. CEA soubor je zdrojový soubor programu ENVIS. CEA soubor je komprimovaný metodou zip a je potřeba dodržovat přesnou strukturu (viz obrázek číslo 1).



Obrázek 1-Struktura CEA archivu

Po rozbalení souboru jsou k dispozici dvě složky, Info a UNI. Složka Info obsahuje XML soubor TreeList a textový soubor VersionHistory. Soubor TreeList obsahuje informace o analyzátoru, ze kterého pocházejí uložená data (viz ukázka kódu číslo 1). Soubor VersionHistory obsahuje informaci o programu, ve kterém byl CEA soubor vytvořen, datum vytvoření a akci. Složka UNI obsahuje strukturu pro univerzální záznamy.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<ObjectList>
	<Object>
		<objekt>DEFAULT</objekt>
		<Identify>
			<BootloaderVersion>37</BootloaderVersion>
			<DeviceAddr>1</DeviceAddr>
			<DeviceNo>525</DeviceNo>
			<DeviceType>13056</DeviceType>
			<HardwareVersion>2</HardwareVersion>
			<PropsType>SMC_Comar</PropsType>
			<SoftwareModules>128</SoftwareModules>
			<SoftwareVersion>3585</SoftwareVersion>
			<Record>
				<record>DEFAULT</record>
				<ID>1</ID>
			</Record>
		</Identify>
	</Object>
</ObjectList>
```

Ukázka kódu 1-Vygenerovaný soubor TreeList

Struktura univerzálních archivů se větví do dvou složek. První s názvem podle ID záznamu a druhá s názvem ArchDefs. Ve složce pojmenované podle ID záznamu se nacházejí složky, jejichž názvy vycházejí z ID archivu. ID jednotlivých archivů jsou pevná a nemění se. Složka se nevytváří, pokud pro daný archiv nejsou stažena žádná data. Složka pojmenovaná podle ID archivu obsahuje dva binární soubory, header a data. Název binárního souboru s daty je určen časem nejstaršího záznamu, například 2015-04-10-08-27-00-041.arch. Zápis do binárního souboru s daty je jednoduchý, obsahuje pouze holé záznamy řazené za sebou. Struktura hlavičky je složitější. Nejprve je zapsána délka hlavičky jako integer, poté jsou zapsány dva longy reprezentující čas prvního a čas posledního záznamu. Následují dva integery, první s počtem záznamů a druhý s periodou. Po těchto základních informacích je ještě nezbytné zapsat časové indexy, které určují mezery mezi záznamy. Ukládají se jako čas a číslo záznamu a rozdělují se na kontinuální a nekontinuální.

Nekontinuální záznamy mají periodu rovnou nule. U kontinuálních se časový index vytvoří, pokud je mezera mezi dvěma časy delší než 1,5 x periody. U nekontinuálních se ukládá časový index u každého x-tého záznamu, kde je x vypočítáno pomocí vzorce 10000 / délka záznamu + 1.

Ve složce ArchDefs jsou uloženy XML soubory obsahující definici ukládaných dat. Název definice dat se skládá z ID archivu a z hashe vypočítaného z obsahu XML souboru, například 00000002-AA14C112.uad. Ukázka obsahu definice dat (viz ukázka kódu číslo 2).

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<UniTypeDescription xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<Props>
        <PropDesc xsi:type="UniPropDesc">
                <PropName>Type</PropName>
                <UserName>Type</UserName>
                <Unit>LOG</Unit>
                <Group/>
                <DataType>Value</DataType>
                <BigEndian>true</BigEndian>
                <mul>1</mul>
                <info>0</info>
                <info2>0</info2>
                <len>2</len>
                <TypeSer>System.UInt16</TypeSer>
        </PropDesc>
</Props>
</UniTypeDescription>
```

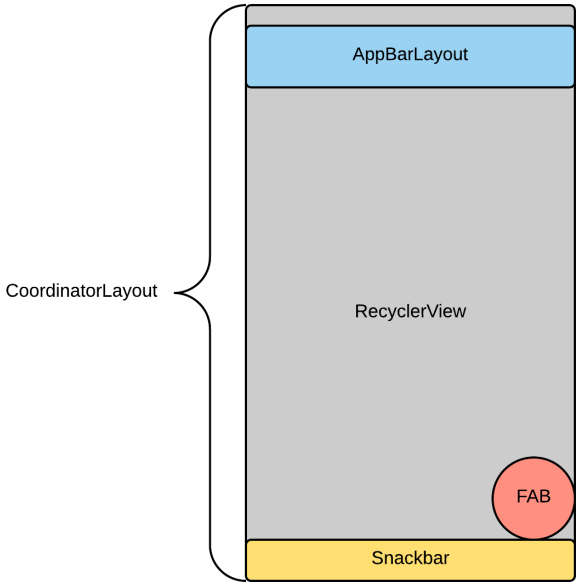
Ukázka kódu 2-Vygenerovaný popis proměnné

3.4 Android

V této části budou popsány nejčastěji používané komponenty, které sloužili jako stavební kameny při vývoji Android aplikace. Aktuálnost komponent byla zajištěna použitím nejnovější Android support knihovny.

3.4.1 CoordinatorLayout

CoordinatorLayout je základní stavební jednotka layoutu, která obaluje celý obsah obrazovky. CoordinatorLayout sám o sobě nedělá mnoho, chová se prakticky jako klasický FrameLayout. Proč ho tedy používat? Protože umožňuje interakci a nastavování chování jeho přímých potomků. Těm je díky připojení nastavení chování umožněno zachytávat událost dotyku, nastavení priority vykreslení, měření layoutu a vnořené scrollování. Nastavení chování se používá k implementaci různých modifikací layoutu, od scrollovatelných layoutů přes layouty, které lze zahodit pomocí pohybu prstem, až k těm, které jsou spolu propojeny. Díky tomu se během scrollu pohybují a animují společně. [9] [10]



Obrázek 2-Rozmístění prvků v CoordinatorLayoutu

3.4.2 AppBarLayout

AppBarLayout je vertikální LinearLayout, který implementuje řadu funkcí z Material Design knihovny, zejména scrollovací gesta. AppBarLayout by měl být potomkem CoordinatorLayoutu, pokud tomu tak nebude, většina jeho funkcí bude nepoužitelná. Přímým potomkům AppBarLayoutu se

nastavuje atribut scroll flags. Tento atribut určuje chování View během události scrollu. Podle něj View reaguje na pohyb, zůstane zobrazené, nebo je odsunuto mimo obrazovku. Často se používá v kombinaci s komponentou RecyclerView. Layout při rozvinutém stavu zobrazuje větší množství informací a během scrollu jsou některé informace skryty. RecyclerView tím získá větší prostor pro zobrazení svého obsahu. [11]

3.4.3 ColapsingToolbarLayout

ColapsingToolbarLayout se používá jako přímý potomek AppBarLayoutu, obaluje Toolbar a rozšiřuje jeho funkčnost. ColapsingToolbarLayout reaguje na scrollování obsahu layoutu a umožňuje měnit velikost a pozici titulku Aktivity v závislosti na stavu layoutu. CollapsingToolbarLayout dědí z FrameLayoutu. Obsahuje Toolbar a další volitelné komponenty. Těmto potomkům lze nastavit chování a docílit tím například paralax efektu. CollapsingToolbarLayout kromě animování potomků poskytuje funkce k změně pozadí a barvy status baru. [12]

3.4.4 Toolbar

Toolbar nahrazuje funkcionalitu ActionBaru, ale na rozdíl od ActionBaru je ViewGroup, může být umístěný kdekoli v layoutu, lze mu nastavit velikost, barvu a vložit do něj další View. Toolbar lze nastavit Aktivitě místo ActionBaru pomocí metody setActionBar. V základu obsahuje navigační tlačítko sloužící jako šipka zpět, zavírací křížek, navigační menu atd. Toolbar může zobrazovat logo v podobě ikonky, titulek a podtitulek, je-li to třeba. Lze do něj umístit jedno nebo více View, jejichž pozice je určena nastavením gravitace. [13]

3.4.5 RecyclerView

RecyclerView je základem při tvorbě listů, nahrazuje již zastaralé ListView. Oproti ListView dává vývojáři mnohem větší volnost při tvorbě složitějších listů s větším počtem rozdílných řádků v listu. S větší volností

pochopitelně přichází složitější implementace. RecyclerView může řadit položky jako list, nebo je umísťovat do mřížky. Typ řazení je určen pomocí LayoutManageru, pro list je to LinearLayoutManager a pro mřížku GridLayoutManager. Směr posouvání je určen orientací, a to vertikální nebo horizontální. Není problém vnořit jednu nebo více instancí RecyclerView do hlavního kontejneru a docílit tím například horizontálního listu umístěného ve vertikálním. RecyclerView se plní pomocí Adaptéru, který v sobě zapouzdřuje funkčnost pro vytváření, recyklaci a kontrolu zobrazení položek. K zobrazení děliče nebo okrajů prvků je třeba implementovat vlastní ItemDecorator. Změnu stavu seznamu lze animovat pomocí standardního ItemAnimatoru, nebo přizpůsobit chování listu pomocí vlastního animátoru. [14]

3.4.6 CardView

CardView je potomek FrameLayoutu. Umožňuje zobrazovat informace v kartách, které mají konzistentní vzhled napříč platformami. CardView má zaoblené rohy a dynamický stín. Na platformě Android 5 a vyšší je toho docíleno díky elevaci, na nižších platformách je stín vytvořen programově. [15]

3.4.7 FloatingActionButton

FloatingActionButton (dále jen FAB) je specifický typ tlačítka, který se používá pro vybrané akce, například přidání položky do listu. FAB je standardně umístěn v pravém dolním rohu layoutu, ale může být na pevno připojený k layoutu. FAB má dvě velikosti, defaultně je nastavena menší. [16]

3.5 Rozbor nejčastěji používaných návrhových vzorů

Návrhové vzory vznikly, protože bylo nutné mít programátorské postupy, které lze označit jako nejvhodnější, usnadňující návrh, zlepšující čitelnost kódu a jeho pozdější modifikaci. Nové návrhové vzory neustále vznikají a nahrazují některé starší, které už jsou vlivem vývoje zastaralé. [17]

3.5.1 Model View Controller

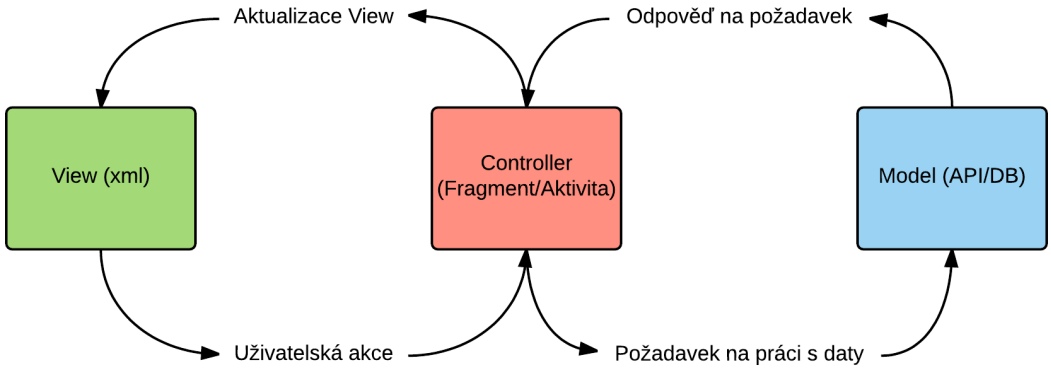
Model View Controller (dále jen MVC) je nejčastěji používaný návrhový vzor v aplikacích pro operační systém Android. MVC umožňuje nejrychlejší způsob vývoje, ale zároveň je nejnevhodnější a nejnáchylnější na chyby.

View je XML layout a reprezentuje to, co vidí uživatel při práci s aplikací.

Model je datová vrstva. Zapouzdřuje operace pro práci s daty, tj. čtení, zápis atd. Nejčastěji bude reprezentován restovým API nebo lokální databází.

Controler je Fragment nebo Aktivita. Obsahuje veškeré metody pro práci s daty, pro zpracování kliknutí, pozorování změny textu atd. Controller je velká nepřehledná třída čítající řádově tisíc i více řádků kódu. Controller je těžké debugovat, provádět na něm testy nebo ho upravovat. Velkou nevýhodou je nutnost ukládat stav při změně konfigurace (Aktivita) a při restartu Aktivity (Aktivita/Fragment).

Mezi hlavní výhody MVC se řadí rychlý vývoj a snadné první kroky. Naproti tomu stojí nevýhody v čele s Controllerem, který je velký nepřehledný objekt, náchylný na chyby s nemožností provádět unit testy. Během vývoje je potřeba ošetřit změny konfigurace a restart Aktivity. Jako poslední nevýhodu lze uvést nevyužití data bindingu. [18]



Obrázek 3-MVC diagram

3.5.2 Model View Presenter

Model View Presenter (dále jen MVP) rozčleňuje aplikaci do tří vrstev, odděluje logiku od Aktivit a Fragmentů, čímž zpřehledňuje kód a umožňuje jednoduše provádět unit testy. Aplikace používající MVP architekturu lze snadno udržovat a rozšiřovat.

View je nejčastěji implementováno pomocí Aktivity nebo Fragmentu. View se stará o inicializace Presentera a drží si na něj referenci. View by mělo být jednoduché, jak jen je to možné, což znemožňuje naplno využít data binding. Jediná jeho funkce je volání metody z Presentera pokaždé, když nastane akce, například kliknutí na tlačítko.

V rámci aplikace pro operační systém Android si lze View představit například jako Aktivitu s XML layoutem. Během inicializace získá instanci Presentera a o nastalých akcích ho informuje pomocí rozhraní. View v sobě také zapouzdřuje veškeré animace.

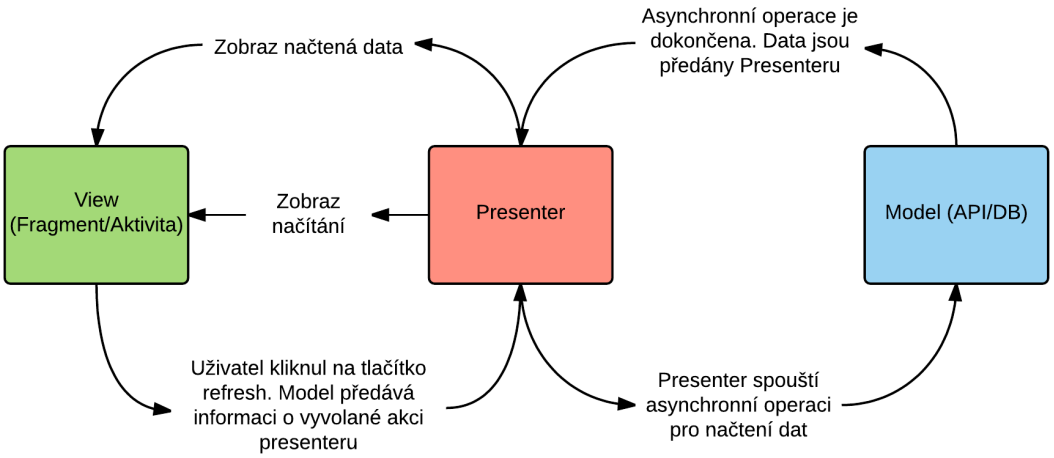
Model je datová vrstva. Model existuje pouze pro manipulaci s daty. Model může být reprezentován například databází nebo API. V případě databáze jde například o DAO (data access object), což je rozhraní pro práci s databází poskytující specifické operace s daty bez odhalení samotné databáze.

Pokud je model reprezentován API, pak ho lze snadno implementovat pomocí knihoven třetích stran, například Retrofit.

Presenter je prostředník mezi Modelem a View. Presenter získává data z Modelu a zformátovaná je předává View. Presenter také zpracovává akce, které nastanou ve View. Veškeré operace na pozadí by měly běžet v Presenteru.

Implementace Presenteru v Androidu může být například pomocí statické proměnné ve View nebo přes dependency injection. Presenter v tomto případě přežije změny konfigurace a restart Aktivity, což výrazně ulehčí vývoj. Operace běžící na pozadí nejsou propojeny s Aktivitou, proto nenastane „memory leak“ a jsou nezávislé na životním cyklu Aktivity.

Hlavní výhodou MVP je rozdělení aplikace do tří nezávislých vrstev, které spolu komunikují pomocí rozhraní. MVP umožňuje snadnou úpravu aplikace, zpřehledňuje kód, který je méně náchylný na chyby, a umožňuje snadné testování aplikace. Jako nevýhody lze uvést delší a náročnější vývoj a nemožnost naplno využít data binding. [19]



Obrázek 4-MVP diagram

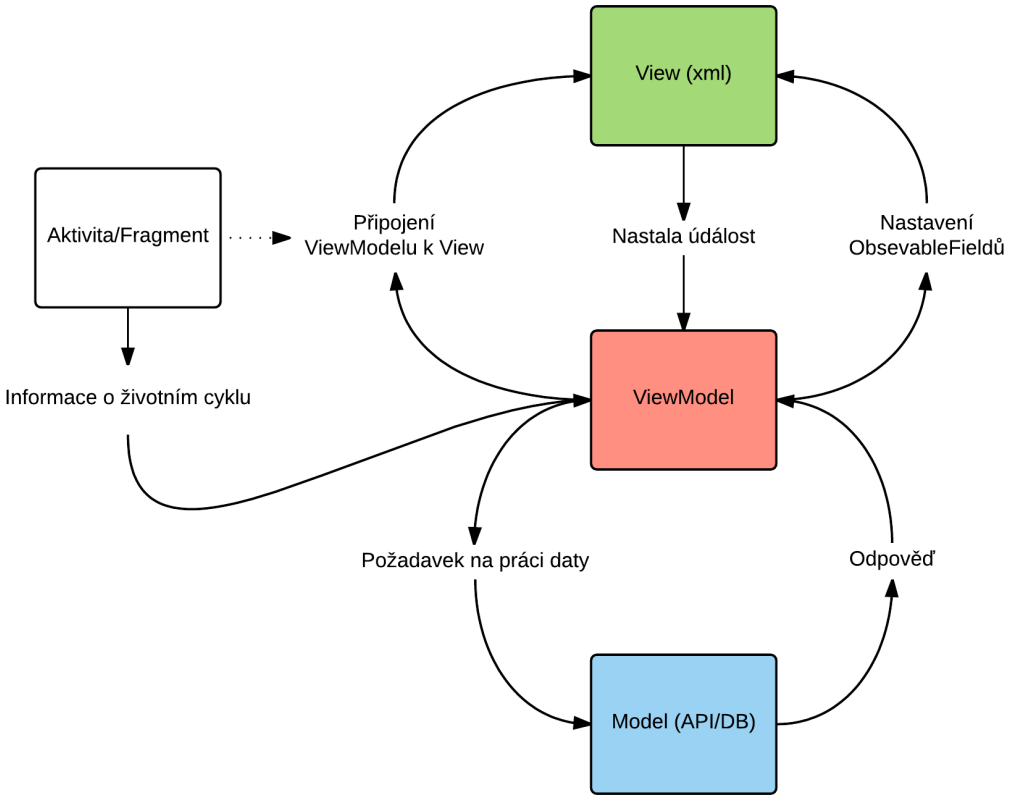
3.5.3 Model View ViewModel

Populární návrhový vzor Model View ViewModel (dále jen MVVM) na Android přichází s uvedením data bindingu. MVVM umožňuje abstrakci stavu a chování View, tím odděluje návrh UI od ovládací logiky. Tento návrhový vzor je vhodný na provádění unit testů, zpřehledňuje kód a usnadňuje modifikaci a rozšiřování aplikace. Návrh aplikace pomocí MVVM vyžaduje porozumění data bindingu, novému přístupu k propojení XML s daty, což může značně prodloužit vývoj aplikace.

Model je podobně jako u MVC a MVP datová vrstva. Umožňuje manipulaci s daty. Může se jednat o restové API nebo lokální databázi.

View je implementováno pomocí XML layoutu. Návrh layoutu je odlišný od MVC a MVP, protože umožňuje provádět základní operace s proměnnými ViewModelu jako kontrolu proměnných, importování objektů a použití jejich metod, proto by se dalo říct, že část logiky je přesunuta do View. Vkládání dat do View je provedeno pomocí sledovatelných proměnných, tzv. ObservableField, jejich změna se automaticky projeví v XML layoutu.

ViewModel je objekt, který reprezentuje stav View. ViewModel implementuje proměnné datového typu ObservableField, které obsahují sledované objekty. Tyto proměnné mohou obsahovat prakticky jakýkoli objekt, může se jednat o textový řetězec, číslo, ale i adaptér pro RecyclerView nebo TextWatcher pro EditText. Pokud je zapotřebí předávat layoutu proměnnou se složitějším datovým typem, tak je nutné napsat binding adapter, který definuje chování layoutu.



Obrázek 5-MVVM diagram

Aktivita nebo Fragment plní funkci připojení ViewModelu k View a předání informací o svém životním cyklu ViewModelu. ViewModel existuje nezávisle na Aktivitě a Fragmentu, proto přežije změnu konfigurace a restart Aktivity. ViewModel drží informace o aktuálním stavu View po celou dobu životního cyklu Aktivity a Fragmentu a je smazán až ve chvíli, kdy jsou Aktivita nebo Fragment kompletně ukončeny.

Mezi hlavní výhody MVVM patří abstrakce stavu View, využití data bindingu, zpřehlednění kódu, vyvarování se zbytečným chybám, snadná úprava aplikace a snadné testování. Nevýhody jsou zdlouhavý vývoj a nutnost znalosti data bindingu. [18]

Pro vývoj aplikace byl vybrán návrhový vzor MVVM. Pro tyto účely byla implementována knihovna Android-ViewModelBinding, která je volně dostupná na githubu. [20]

4 Praktická část

Praktická část popisuje vývoj samotné aplikace. Nejprve je rozebrána knihovna, která zajišťuje komunikaci s přístrojem a vytvoření CEA souboru, a následuje popis aplikace využívající výše zmíněnou knihovnu.

4.1 Datové struktury

Datové struktury jsou objekty, které spolu dohromady tvoří celek, který zajišťuje správnou reprezentaci stažených dat. Z holých dat vytváří instance konfigurací, záznamů, aktuálních dat atd.

4.1.1 ByteArray

Třída ByteArray byla navržena pro zjednodušení zápisu a čtení dat. Její hlavní funkčnost je převod pole bajtů na jednoduché a složené datové typy a naopak. Kromě čtení a zápisu dat v sobě také zapouzdřuje kontrolu validity pomocí CRC součtu.

Třída ByteArray obsahuje pole bajtů a čítač. V poli mohou být data načtená z přístroje, nebo se může jednat o prázdné pole, do kterého budou data zapisována. Data v poli lze uložit do CEA archivu, nebo je lze odeslat přístroji ve zprávě. Čítač si drží aktuální pozici v poli, do které lze zapisovat, nebo z ní číst. Jeho inkrementace probíhá automaticky. Pozici pro čtení či zápis dat lze zvolit i manuálně bez toho, aby byl čítač ovlivněn.

Klíčovou funkčností třídy ByteArray je možnost data číst i zapisovat jako BigEndian(nejvíce významný bajt první) a LittleEndian(nejméně významný bajt první). Pro čtení a zápis byly navrženy funkce využívající bitové rotace a logického součtu. Funkce umožňují čtení a zápis celé škály proměnných. Z důvodu chybějící podpory bezznaménkových datových typů v programovacím jazyce Java bylo nutné každou funkci ošetřit tak, aby proměnná na výstupu funkce byla validní. Kromě jednoduchých datových typů

třída ByteArray pracuje i se složitými datovými typy. Jedná se o čtení a zápis polí bajtů a práci s třídou Calendar pro čtení a zápis času.

Třída ByteArray dále umožňuje součet dvou instancí ByteArray, kontrolu null, kontrolu délky a generaci kontrolního součtu CRC. Součet se používá při ověření validity dat, která se provádí před odesíláním a po přijetí dat. Dále třída umožňuje převod textových řetězců na pole bajtů a převod mezi BCD a Dec.

4.1.2 Universal Configs

Třída UniversalConfigs obsahuje nastavení potřebná pro správnou interpretaci dat z analyzátoru, nastavení archivů a nastavení samotného analyzátoru.

Během inicializace třídy UniversalConfigs je vyžadována instance třídy DeviceManager, která se používá pro načtení konfigurací z přístroje. Pro načtení dat byla implementována metoda UpdateConfig, která načítá a ukládá požadované nastavení. Pro rozlišení jednotlivých konfigurací byly navrženy enumy ERezim a EDataType. Data konfigurací jsou načítána jako pole bajtů a zpracována pomocí třídy ByteArray. Konfigurace jsou inicializovány pomocí metody GetData, která pomocí výše zmíněných enumů vyhodnocuje, o kterou konkrétní konfiguraci se jedná.

Konfigurace jsou důležité při vytváření univerzálních archivů ze stažených dat. Konfigurace obsahují nastavení jednotlivých archivů, například to, které proměnné jsou zaznamenávány, pod jakým jménem, jakou mají veličinu a v jakém datovém typu jsou ukládány. Tyto informace jsou velmi důležité při vytváření CEA archivu, bez jejich správného zapsání by nebylo možné požadované hodnoty správně načíst a zobrazit.

Konfigurace je ukládána co nejuniverzálněji, proto data nejsou ukládána ve složitých datových strukturách s mnoha proměnnými. Bylo navrženo několik jednoduchých tříd, které obsahují vždy jednu proměnnou a informace pro správnou interpretaci dané proměnné. Instance těchto tříd jsou uchovávány v separovaných listech podle toho, zdali se jedná o konfiguraci archivu nebo přístroje. Třída UniversalConfigs se skládá z několika listů obsahujících instance objektů s proměnnou a jejím popisem. Jedná se o třídy InfoConfig, InfoArchiveC, InfoData a InfoArchiveD.

Výše zmíněné třídy obsahují id konfigurace, název konfigurace a informaci o změně konfigurace. Mohou v sobě zapouzdřovat instanci další konfigurace, čímž lze vytvářet struktury dat. Třídy InfoData a InfoArchiveD obsahují proměnnou a její popis. Hodnota je uložena jako objekt a její datový typ je určen pomocí enumu ETyp. Třídy InfoConfig a InfoArchiveC uchovávají proměnnou v datovém typu integer.

Pro zjednodušení čtení a zápisu proměnných byly ve třídě UniversalConfigs navrženy metody GetValue a SetValue. Metoda SetValue přebírá id konfigurace, název proměnné, proměnnou a datový typ proměnné. Metoda vyhledá v listu načtených konfigurací požadovanou konfiguraci, uloží do ní hodnotu v zadaném datovém typu a nastaví v konfiguraci informaci o změně. Metoda GetValue pomocí id a názvu proměnné vyhledá proměnnou a převede ji do požadovaného datového typu. V případě nenalezení požadované proměnné metoda přebírá instanci třídy Object, ve které je uložena defaultní hodnota, která bude vrácena.

Důležitou roli při vytváření CEA archivů má třída ArchiveProcessor, která na základě konfigurací a typu archivu načte z konfigurací informace potřebné pro vytvoření CEA archivu. Jedná se například o id archivu, periodu ukládání dat nebo verzi archivu.

4.1.3 PropDesc

Třída PropDesc je základní třída pro popis proměnné. Zapouzdřuje v sobě název proměnné, datovou jednotku, skupinu proměnné, podskupinu proměnné a informaci, jestli jde o jednoduchou proměnnou nebo čítač. Z třídy PropDesc dědí třídy UniPropDesc a VirtualPropDesc.

4.1.4 UniPropDesc

Třída UniPropDesc se používá pro popis univerzální proměnné. Třída UniPropDesc je potomkem třídy PropDesc. Oproti svému předkovi obsahuje informaci o tom, zdali jde o datový typ BigEndian. Pokud popisovaná proměnná je pole, tak obsahuje jeho délku.

Hlavní metodou třídy UniPropDesc je GetLenAndSet, která podle nastaveného datového typu proměnné vypočítá délku proměnné jako počet bajtů, nastavuje návratový typ a ukládá textový řetězec, který obsahuje datový typ proměnné v jazyce C#, aby bylo možné proměnnou načíst v programu ENVIS.

Uložení popisu proměnné je provedeno ve formátu XML. XML popis je generován pomocí metody Serialize. Popis proměnné je uložen při vytváření archivu. Metodě se v parametrech předává instance třídy Document a instance třídy Element sloužící k vytvoření XML souboru.

Pro zjednodušení vytvoření elementů reprezentujících popis datového typu byla implementována metoda createElement. Metoda createElement vrací element vytvořený s využitím instance třídy Document a dvou textových řetězců obsahujících název a hodnotu elementu. Objekty se metodě předávají jako parametry.

4.1.5 SmpActData

Třída SmpActData složí k získání aktuálních dat z přístroje. Zapouzdřuje v sobě metody pro vytvoření požadavku na aktuální data, jeho odeslání a následné zpracování odpovědi. Konstruktoru třídy se přidávají instance objektů DeviceManager a UniversalConfigs. DeviceManager je použit pro komunikaci se zařízením. UniversalConfigs se používá pro správnou interpretaci přijatých dat. Třída disponuje metodami UpdateUNIConfig a UpdateUNIData. První metoda slouží pro načtení konfigurace a druhá pro načtení dat. Metoda pro načtení konfigurace je spuštěna vždy před načtením aktuálních dat, pokud konfigurace již nebyla načtena. Frekvence obnovy aktuálních dat je určena frekvencí volání metody UpdateUNIData. Veškerá načtená data jsou přístupná přes veřejné proměnné.

4.1.6 ArchiveInformation

Třída ArchiveInformation obsahuje informace o počtu záznamů. Kolik je aktuálně zapsaných záznamů, maximální počet záznamů, adresu prvního záznamu a informaci, zdali přístroj přepisuje záznamy od začátku. K přepisování záznamů od začátku dojde v případě, že již přístroj vyčerpal celou paměť určenou pro daný archiv.

Inicializace třídy ArchiveInformation probíhá pomocí ByteArray daného archivu s délkou nula. Při odeslání dotazu na záznamy s počtem nula odpoví přístroj pouze hlavičkou, v které jsou uloženy potřebné informace. Hlavička se skládá z CRC součtu, typu archivu, maximálního počtu záznamů a aktuálního záznamu.

4.1.7 KMBTime a KMBString

Třída KMBTime umožňuje převádět čas uložený v ByteArray na zobrazitelný čas. Přístroj vrací veškeré časové údaje jako počet milisekund od 1.1.2000 00:00:00. Třída pro konverzi času obsahuje metody timeToMilliseconds

a milisecondsToTime, které zajišťují převod mezi časem generovaným analyzátorem a mezi časem určeným k zobrazení. Třída KMBTime umožňuje také převod mezi letním a zimním časem.

Třída KMBString zajišťuje jednoduchý převod mezi polem bajtů a textovým řetězcem.

4.2 Komunikační struktury

Komunikační struktury zaštiťují komunikaci s přístrojem, odeslání požadavku, přijetí odpovědi a nastavování přístroje. Díky těmto strukturám je snadné vytvořit validní zprávu a následně ji odeslat.

4.2.1 MessageKmbLong

Třída MessageKmbLong byla navržena pro zjednodušení vytváření zpráv pro analyzátor. Jde o zprávy s požadavkem na data, změnu nastavení, stažení konfigurací analyzátoru nebo resetování analyzátoru. Třída MessageKmbLong má předka Message. Třída Message obsahuje předdefinované bajtové konstanty, podle kterých přístroj pozná, jaký požadavek mu byl zaslán. Třída MessageKmbLong obsahuje statické metody pro generování zpráv a metody pro generování CRC součtu.

4.2.2 DeviceManager

Třída DeviceManager reprezentuje zařízení. Zapouzdřuje v sobě informace potřebné pro komunikaci, informace o analyzátoru a informace o konfiguraci analyzátoru. Obsahuje metody pro odesílání požadavků a přijímání odpovědí.

Konstruktoru třídy DeviceManager se předává String s URL přístroje a číslo portu. Po vytvoření instance je možné začít komunikaci s analyzátorem.

Třída také obsahuje univerzální konfiguraci přístroje a univerzální konfiguraci archivů uložené jako dvě instance třídy UniversalConfigs.

Konfigurace se automaticky inicializují během načítání dat. Pokud jsou vyžadovány konfigurace během načítání dat, tak jsou automaticky načteny, použity a uloženy.

Komunikace s analyzátorem probíhá pomocí privátní metody requestAndAnswer. Tato metoda má parametr ByteArray reprezentující zprávu k poslání a vrací přijatou odpověď jako instanci třídy ByteArray. Jak napovídá název metody, během komunikace se odešle požadavek, přijme odpověď a ukončí se připojení. Komunikace probíhá pomocí tříd Socket, DataOutputStream a DataInputStream. Po dokončení komunikace jsou streamy a socket uzavřeny. Bylo nutné zajistit, aby se streamy vždy uzavřely a nezabíraly připojení k přístroji. Soustavné neuzavírání připojení by vedlo k přehlcení a nedostupnosti přístroje na 24 hodin. Poté přístroj sám automaticky ukončí neuzavřená připojení. Metoda requestAndAnswer kromě odeslání a přijetí dat zajišťuje také validitu obdržených dat. Validita se kontroluje pomocí CRC součtu.

Pro vytvoření požadavku a přijetí odpovědi byla ve třídě DeviceManager vytvořena přetížená metoda GetData. Její první varianta má parametr MessageKmbLong a návratovou hodnotu v podobě instance třídy ByteArray. Druhá varianta je navržena pro načtení univerzálních konfigurací. Její parametry jsou UniversalConfigsMessage a návratový typ. Tato metoda vrací konfiguraci jako instanci třídy předané v parametru metody.

Kromě zaznamenaných dat a konfigurací zařízení lze načíst informace o analyzátoru. Pro tento účel byla implementována metoda loadIdentification. Metoda loadIdentification odešle dotaz na identifikaci zařízení a odpověď uloží do lokálních instancí objektů SmpNotebook a SmpIdentify. Třída SmpNotebook obsahuje název objektu a název měření. Třída SmpNotebook obsahuje informace o zařízení a to číslo zařízení, verzi softwaru, verzi

hardwaru, softwarové moduly, adresu zařízení, verzi bootloaderu, typ zařízení a vlastnosti zařízení. Tyto informace jsou využity při vytváření souboru TreeList.XML.

4.3 Tvorba CEA souboru

Po stažení a uložení dat do objektů je nutné tyto data zapsat do souborů a uložit je do přenosného CEA souboru. K tomu byly implementovány třídy popsané níže. Tyto třídy zajišťují validitu a správnou strukturu výsledného souboru.

4.3.1 ArchiveBuilderHolder

Během vytváření stromové struktury archivu je potřeba vrátit více instancí objektů, ale Java umožňuje jako výsledek metody vracet pouze jeden objekt. K překonání tohoto problému byla navržena třída ArchiveBuilderHolder. Třída ArchiveBuilderHolder obsahuje tři veřejné objekty: archiveFile, archiveHeaderFile a headerBuilder. Objekty archiveFile a archiveHeaderFile reprezentují instance třídy File, určené pro zápis dat. Zápis do těchto souborů probíhá po stažení záznamů. Během postupného stahování záznamů probíhá aktualizace hlavičky pomocí metod třídy HeaderBuilder.

4.3.2 ArchiveBuilder

Pro vytvoření struktury CEA souboru a zápis dat byla navržena třída ArchiveBuilder. Její použití je jednoduché a pomocí jejích metod lze snadno zapsat data pro jakýkoli typ archivu. Obsahuje dvě instance třídy File: archRecordsFile a archDefsFile. Jejich vytvoření je zajištěno pomocí metody createStructure. Metoda createStructure vytváří strukturu složek určenou pro zápis dat. Metoda má dva parametry v podobě textových řetězců. Jeden obsahuje místo, kde bude datová struktura vytvořena. Druhý textový řetězec obsahuje informace, které budou zapsány do souboru TreeList.XML. Tento

soubor je vytvořen již při inicializaci a v průběhu zápisu dat se nemění. Druhým souborem vygenerovaným při tvorbě struktury je VersionHistory.txt.

Pro zjednodušení vytváření složek byla navržena privátní metoda createFile. Metoda createFile vrací instanci třídy File. Parametrem této funkce je instance třídy File udávající složku, do které bude nová instance třídy File vytvořena, a název nové složky jako textový řetězec.

Pro zjednodušení zápisu do souborů byly implementovány dvě statické metody: writeByteArray a addByteArray. Obě metody mají jako parametry instanci třídy File a pole bajtů. Metoda writeByteArray provede zápis bajtů do souboru a uzavře ho. Metoda addByteArray připíše bajty na konec souboru a uzavře ho.

Data určená k zápisu v podobě pole bajtů jsou získávána pomocí statické metody getArchiveData. Metoda getArchiveData má parametr ByteArray. Instance třídy ByteArray obsahuje surová data, která byla přijata od analyzátoru. Od těchto dat je třeba odříznout hlavičku o velikosti 15 bajtů.

Vytvoření souborů určených pro zápis dat je provedeno pomocí metody createArchiveFile. Metoda createArchiveFile přebírá jako parametr data určená k zapsání jako instanci třídy ByteArray, textový řetězec UTD, instanci třídy ArchiveProcessor a ID archivu. Textový řetězec UTD obsahuje formát univerzálních dat, který je zapsán do souboru ve složce ArchDefs. Instance třídy ArchiveProcessor je využita při vytváření pole bajtů pro zápis. ID určuje název souboru definujícího data a používá se při vytváření instance třídy HeaderBuilder. Volání metody createArchiveFile vytvoří zbytek struktury. Struktura se liší podle typu archivu. Metoda createArchiveFile má návratovou proměnnou typu ArchiveBuilderHolder, která obsahuje archiv a hlavičku archivu, jako instance třídy File, a instanci HeaderBuilder. Tyto proměnné mohou být použity pro zapisování dalších záznamů z analyzátoru. Data budou

zapisována v případě stahování většího množství záznamů tak, že přístroj nebude schopný odeslat požadovaný počet v jednom dotazu.

4.3.3 ArchiveBuilderOptions

Třída ArchiveBuilderOptions zapouzdřuje nastavení stažení archivu. Obsahuje tři proměnné, které se inicializují pomocí konstruktoru. První proměnná je typ archivu uložený jako EArchiveType. Zbylé proměnné udávají počáteční adresu, ze které budou stahována data, a počet záznamů ke stažení.

4.3.4 ArchiveBuilderInterface

Interface ArchiveBuilderInterface je určený k informování o průběhu vytváření CEA archivu. ArchiveBuilderInterface obsahuje veřejnou metodu onProgressChange. Metoda má parametry progress a archiveType. Parametr progress je integer udávající stav vytváření archivu. Parametr archiveType obsahuje typ archivu, na kterém se právě pracuje.

4.3.5 HeaderBuilder

Třída HeaderBuilder byla navržena k vytváření hlavičky archivů. Třída HeaderBuilder je univerzální pro všechny typy archivů a její instanci je potřeba držet během celého zapisování archivu. Pomocí třídy HeaderBuilder probíhá postupné aktualizování hlavičky v průběhu přijímání záznamů, které jsou zapisovány do archivu. Do hlavičky jsou zapisovány časy jednotlivých záznamů pro ulehčení vyhledávání. Po přijetí posledního záznamu a zapsání času je hlavička kompletní a může následovat její zapsání do souboru.

Inicializace třídy probíhá pomocí konstruktoru s třemi parametry. Parametry konstruktoru jsou pole bajtů reprezentující první dávku záznamů, instanci třídy ArchiveProcessor a ID archivu. Z pole bajtů je vytvořena instance třídy ByteArray. Pro práci s touto instancí byly navrženy čtyři metody: getFirstRecordMillis, addByteArray, getTimes a getByteArray.

Metoda getFirstRecordMillis vrací počet milisekund prvního záznamu. Tato hodnota je vyžadována při vytváření názvu archivu.

Pro případy, kdy je nutné stahovat záznamy po dávkách, byla naimplementována metoda addByteArray. Metoda má parametr pole bajtů. Data jsou zkontrolována a je z nich vytvořena instance třídy ByteArray. Tato nově vzniklá instance je následně připojena k již existující instanci třídy ByteArray, která byla vytvořena při inicializaci třídy HeaderBuilder.

Metoda getTimes vrací pole longů. Pole longů obsahuje časy jednotlivých záznamů v podobě milisekund. Záznamy jsou načítány z instance třídy ByteArray a u každého záznamu je ze správné pozice přečtený čas. Pozice je získána pomocí instance třídy ArchiveProcessor a její proměnné RemBytes. K získání délky záznamu je opět využita instance třídy ArchiveProcessor.

Metoda getByteArray byla implementována k získání dat určených pro zápis do souboru header.bin. Metodu lze spustit až ve chvíli, kdy byl načten poslední záznam z analyzátoru. Z dat, která metoda vrací, lze vytvořit hlavičku pro jeden konkrétní typ záznamů. Metoda getByteArray kromě vytváření dat pro hlavičku také počítá časové indexy. Časové indexy se dělí na kontinuální a nekontinuální podle typu archivu. Pro proměnou časového indexu byla navržena třída TimeIndex. Třída TimeIndex má dvě hodnoty: index času a samotnou časovou hodnotu v podobě milisekund. Jednotlivé časové indexy se ukládají do listu. A po uložení posledního jsou vybrané indexy zapsány do hlavičky kvůli zjednodušení vyhledávání v záznamech.

4.3.6 ZipBuilder

Třída ZipBuilder slouží ke komprimaci souborové struktury vytvořené pomocí třídy ArchiveBuilder, umožňuje vytvořit CEA soubor. Třída ZipBuilder implementuje jednu statickou veřejnou metodu zipFiles. Metoda zipFiles má parametr typu File a textový řetězec. První parametr je složka obsahující

soubory, které mají být zkomprimovány. Druhý parametr reprezentuje cestu k výslednému CEA souboru. K usnadnění vytvoření CEA souboru byly implementovány tři statické metody: zipFolderArray, addFileToZip a addFolderToZip.

Metoda zipFolderArray prochází zdrojovou složku a zazipovává všechny podsložky do výsledného CEA souboru.

Metoda addFileToZip přidává do výstupního komprimovaného souboru soubor, do kterého jsou zapsána data předaná této metodě jako parametr.

Metoda addFolderToZip byla navržena pro přidání složky do výstupního komprimovaného souboru. Metoda addFolderToZip je rekurzivní metoda, která prochází složku, jež jí byla předána jako parametr. Pokud složka obsahuje podsložky, tak pro každou složku je rekurzivně volána metoda addFolderToZip. V případě nalezení souboru je soubor vytvořen pomocí metody addFileToZip.

4.3.7 CEAArchive

Třída CEAArchive byla naimplementována jako hlavní nástroj pro vytváření CEA souboru. Konstruktor třídy CEAArchive má dva parametry. První parametr je instance třídy DeviceManager. Druhý parametr je cesta, kam má být CEA soubor vytvořený pomocí knihovny uložen. Třída disponuje veřejnou přetíženou metodou createCEAArchive.

Metoda createCEAArchive byla implementována jako spouštěč vytváření CEA souboru. Má vždy parametr archiveOptionsList. ArchiveOptionsList je list, který obsahuje objekty typu ArchiveBuilderOptions. Každé jedno nastavení v sobě obsahuje informaci o typu archivu ke stažení, počáteční adresu a počet záznamů. Metoda createCEAArchive má volitelný parametr typu ArchiveBuilderInterface, který je vhodné naimplementovat

v případě sledování průběhu vytváření CEA souboru. Při zavolání metody createCEAArchive je stažena identifikace analyzátoru, konfigurace archivů a konfigurace přístroje. Následuje vytvoření instance třídy ArchiveBuilder a vytvoření struktury CEA souboru.

Po vytvoření struktury se prochází listem s nastavením archivů. Pro každé nastavení je vytvořena instance třídy ArchiveProcessor. Instance třídy ArchiveProcessor se inicializuje pomocí metody getArchiveProcessor. Následuje stažení konkrétních dat, vytvoření souborů a instance třídy HeaderBuilder pomocí metody createArchiveFile z třídy ArchiveBuilder. Tyto objekty mohou být využity pro zápis dalších dat, pokud není možné stáhnout celý archiv najednou. Po stažení všech dat je zapsána hlavička pomocí statické metody writeByteArray třídy ArchiveBuilder.

Ve chvíli, kdy jsou stažena a zapsána data, je celá stromová struktura komprimována pomocí metody zipFiles třídy ZipBuilder do CEA souboru.

4.4 Návrh aplikace

Pro vytvoření mobilní aplikace byl zvolen návrhový vzor Model View ViewModel, protože klasický návrhový vzor Model View Controller je nevhodný pro komplexnější aplikace. Umožňuje rychlý vývoj, ale následná úprava aplikace je náročnější a s narůstajícím kódem narůstá i nepřehlednost. Rozhodování mezi MVVM a View Model Presenter bylo složitější, ale MVP nevyužívá celý potenciál data bindingu, proto byl zvolen návrhový vzor MVVM.

4.4.1 Databáze zařízení

Databáze byla implementována z důvodu nutnosti ukládat základní informace, jako jsou název, adresa, port a popis zařízení. Databáze je bez relačních vztahů, určená pouze pro ukládání, mazání a modifikaci analyzátorů, se kterými pracuje uživatel.

Operační systém Android umožňuje ukládat data do SQLLite databáze, ale práce s touto databází je zdlouhavá. Proto byla k urychlení návrhu použita knihovna ActiveAndroid, která k zjednodušení návrhu tabulek využívá reflexi.

Pro přístup a práci s daty byl implementován DAO objekt, který zapouzdřuje veškeré operace prováděné na tabulce se zařízeními. Metody DAO objektu lze spustit přímo ve hlavním vlákně, ale to by mohlo vést ke zpomalení aplikace. Z tohoto důvodu byly implementovány dotazy. Každý dotaz reprezentuje jednu konkrétní činnost prováděnou s daty. Například dotazy pro vytvoření záznamu, pro změnu, pro čtení, pro smazání atd. Všechny dotazy mají společného abstraktního předka, třídu Query (viz ukázka kódu číslo 3).

```
public abstract class Query
{
    private Bundle mMetaData = null;

    public abstract Data<?> processData();

    public Bundle getMetaData()
    {
        return mMetaData;
    }

    public void setMetaData(Bundle metaData)
    {
        mMetaData = metaData;
    }
}
```

Ukázka kódu 3-Abstraktní třída Query

Spouštění dotazů probíhá pomocí třídy DatabaseCallManager. Jmenovaná třída byla navržena jako mechanismus pro správu dotazů, přes které se přistupuje k databázi. Umožňuje spustit dotaz, zrušit dotaz nebo zjistit, zdali daný dotaz aktuálně neprobíhá. Dotazy jsou prováděny na pozadí, čímž je zajištěna plynulost aplikace při zpracování většího množství dat. Po dokončení

dotazu jsou data vrácena pomocí rozhraní. Pokud během provádění dotazu dojde k chybě, jsou vráceny informace o chybě.

4.4.2 Animace

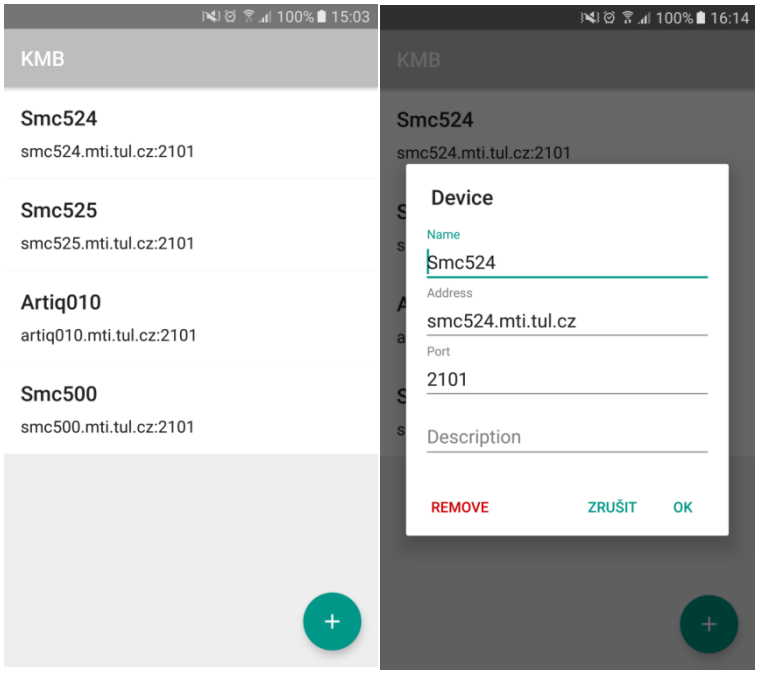
Během přechodu z jedné Aktivity do druhé lze sledovat plynulou animaci v podobě přijetí nové Aktivity ze strany displeje. Aktivita nepřijíždí celá, ale „rozřezaná“ na části. Toho bylo docíleno správným návrhem layoutu a použitím CoordinatorLayoutu. Aktivitě je nastavena animace během spouštění pouze tehdy, pokud se jedná o verzi Androidu Lollipop a vyšší, protože nižší verze systému nejsou podporovány. Jednotlivé elementy plynule přijedou na displej a složí se v jeden celek.

4.4.3 Hlavní obrazovka

Hlavní obrazovka byla navržena jako seznam analyzátorů. Může se nacházet ve třech základních stavech, zobrazení obsahu, načítání nebo prázdný seznam. Prázdný seznam je zobrazen ve chvíli, kdy dotaz na databázi se zařízeními vrátí pole o velikosti nula, protože nebyly přidány žádné analyzátory, uživatel je informován obrazovkou s textovou zprávou. Layout načítání se zobrazuje v případě, že uživatel zapnul aplikaci a je třeba provést dotaz na databázi k získání dat. Po získání dat je zobrazena buď prázdná obrazovka, nebo seznam se zařízeními.

Pro implementaci seznamu byl zvolen RecyclerView. Je to složitější řešení oproti klasickému ListView, ale RecyclerView je vhodnějším kandidátem vzhledem k nutnosti zobrazovat animace editace, přidávání a mazání položek. ListView by v případě změny překresloval celý list a průběh tohoto procesu se projevuje nepříjemným bliknutím. RecyclerView odděluje položky jemnou linkou pomocí předdefinované dekorace. O správné řazení dat se stará vertikální LinearLayoutManager.

K naplnění RecyclerView byl navržen adaptér. Ten umožňuje zobrazovat pouze jeden typ položky, jejíž vzhled je nadefinován pomocí XML layoutu. Kromě XML layoutu byl implementován ViewHolder, který se používá během recyklace položek, a díky němu nemusí být layout pokaždé načítán znovu. Kromě toho také definuje událost krátkého a dlouhého kliknutí. Naplnění položky daty probíhá spuštěním metody bindData zmíněného ViewHolderu, které se předává entita držící informace o analyzátoru. Entita je použita při vytváření ViewModelu, který je přes data binding nastaven layoutu. Adaptér v sobě zapouzdřuje metody pro přidání, nastavení nebo odebrání položky. Metody pracují s listem a notifikují RecyclerView o tom, na jaké pozici probíhá změna. RecyclerView změnu provede pomocí animace.



Obrázek 6-Hlavní Aktivita a dialog pro práci se zařízením

Přidávání zařízení bylo implementováno pomocí FAB a dialogu. Dialog slouží také pro editaci a odebrání záznamu. Jeho zobrazení je zahájeno po dlouhém stisku položky, nebo po kliknutí na FAB. Tlačítko pro odebrání je pochopitelně skryto, pokud uživatel vytváří nové zařízení.

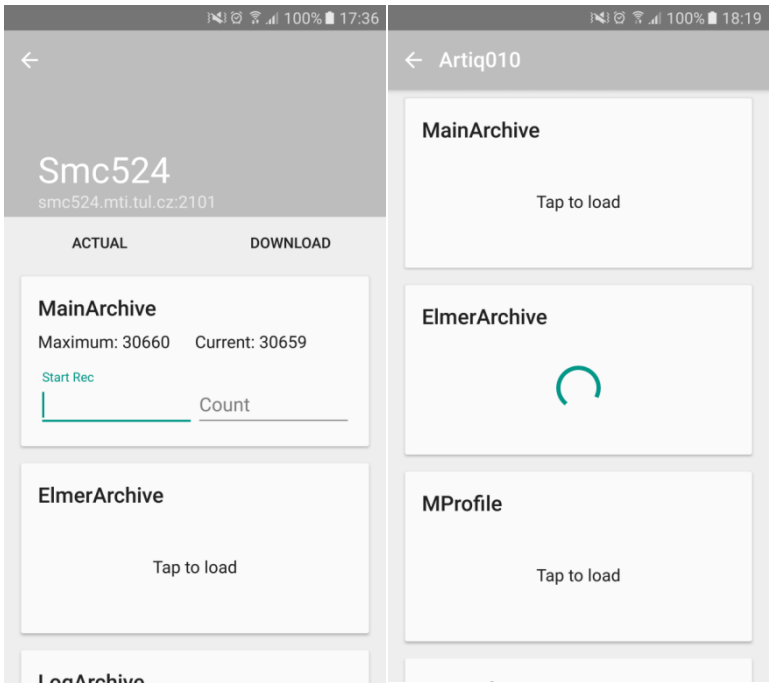
Dialog pro přidávání a editaci zařízení byl navržen pomocí jednoduchého XML layoutu, který obsahuje čtyři TextInputLayouty s EditTextem umístěné pod sebou. TextInputLayout slouží k animaci nápovědy. V EditTextu je zobrazena nápověda, pokud neobsahuje žádný text. Ta je po kliknutí pomocí animace zmenšena a odsunuta nad EditText, kde zůstane, pokud byl zadán text. TextInputLayout zpřehledňuje vyplňování informací a byl použit, protože uživatel díky němu ví, jaké informace právě vyplňuje. Dialog byl navržen tak, aby při potvrzení vytvoření nového záznamu proběhla validace. Pokud validace neproběhne v pořádku, dialog se nezavře a uživatel je informován o zjištěné chybě.

Zařízení je po vytvoření zobrazeno na konci seznamu, protože se řadí podle data vytvoření vzestupně. Zařízení lze editovat či smazat po dlouhém stisku položky, po krátkém stisku položky je spuštěna Aktivita s detailem zařízení.

4.4.4 Detail zařízení

Detail se vyznačuje výrazným CollapsingToolbarLayoutem, pod kterým jsou vertikálně řazeny ovládací prvky a karty. Detail zobrazuje informace uložené v lokální databázi a umožňuje stažení aktuálního stavu paměti pro konkrétní záznam, který analyzátor ukládá. Tyto informace mohou být použity při spouštění služby pro stažení archivu.

CollapsingToolbarLayout jako titulek používá název zařízení, podtitulek je adresa zařízení a port. Během scrollování je layout zabalen a podtitulek skryt pomocí paralax efektu, čímž je získáno více místa pro hlavní obsah. Celý proces probíhá naprosto plynule, titulek je postupně zmenšován a přemístěn do Toolbaru. CollapsingToolbarLayout je obalený AppBarLayoutem, pod kterým se nachází NestedScrollView zobrazující hlavní obsah detailu.



Obrázek 7-Detail zařízení

V horní části obsahu jsou umístěna ovládací tlačítka pro zobrazení aktuálních dat a pro stažení archivu. Pod nimi se nacházejí jednotlivé položky pro stažení aktuálního stavu záznamů. Položky jsou reprezentovány moduly, kterým se nastavuje typ archivu. V závislosti na typu se nastavuje titulek a data pro stažení. Uživatel si po spuštění Aktivity pomocí kliknutí načte archivy, s kterými chce dál pracovat.

Modul pro práci s archivem je prosté View, kterému se přiřadí ViewModel. Byl navržen jako karta, která se vyznačuje zaoblenými rohy a jemnou elevací. Samotný layout se skládá z CardView, které obsahuje LinearLayout s titulkem a FrameLayoutem. FrameLayout slouží k zobrazení tří stavů, kterých může View nabývat. Základní je prázdný stav, který vybízí uživatele, aby kliknutím načetl data. Druhý stav zobrazuje průběh. Třetí stav informuje uživatele o maximálním počtu záznamů a posledním záznamu. Pod těmito informacemi jsou dva EditTexty obalené TextInputLayoutem, do nich se zadává první záznam, který má být stažen a počet následujících záznamů.

Po zvolení archivů a zadání informací potřebných pro stažení lze kliknutím na tlačítko Download zahájit stahování. Před samotným spuštěním procesu stahování probíhá validace zadaných dat a v případě chybně zvolených intervalů je uživatel informován o problému. Pokud kontrola proběhne úspěšně, je spuštěna služba, která stáhne data na pozadí.

4.4.5 Služba pro stažení CEA archivu

Stažení a vytvoření CEA archivu probíhá v závislosti na zvoleném intervalu a počtu archivů. Aby aplikace nebyla blokovaná stahováním archivu, byla implementována služba. Služba je jednorázová, spustí se, provede stažení a potom ukončí svůj životní cyklus. Předává se jí entita se zařízením, ke kterému se bude připojovat, a nastavení intervalů jednotlivých archivů. Služba blokuje přechod CPU do úsporného režimu, aby nedošlo k vypnutí procesu během uspání mobilu. Po dokončení činnosti je blokování ukončeno a služba se vypne.

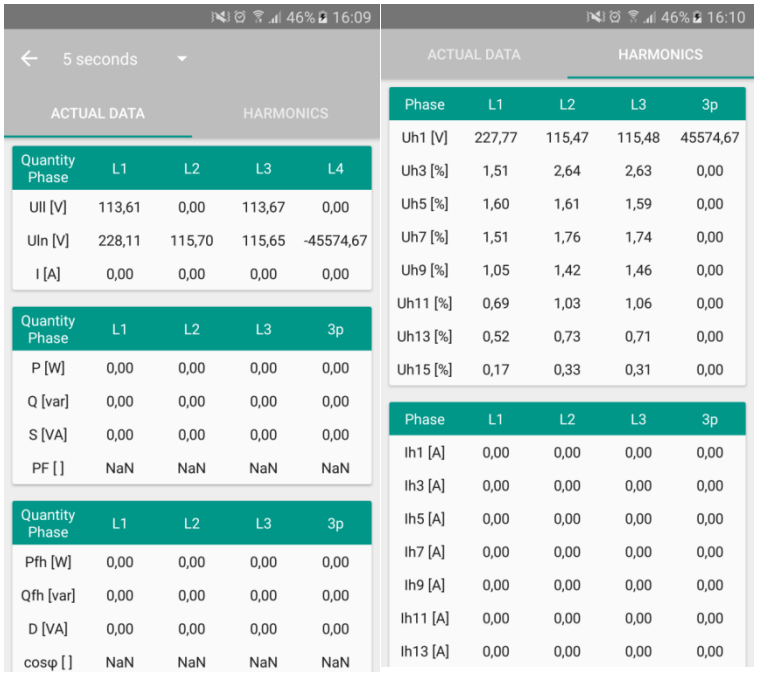
Uživatel je o průběhu stahování informován pomocí notifikace. Notifikace nabývá třech stavů. První je procentuální průběh, zobrazený během stahování. Druhé dva stavy informují uživatele o úspěšném, či neúspěšném stažení archivu.

4.4.6 Aktuální data

Obrazovka s aktuálními daty dává uživateli přehled o tom, co se právě v analyzátoru děje. Zobrazuje velké množství dat, proto byl k implementaci využit ViewPager. ViewPager umožňuje zobrazovat data v tabech, mezi kterými se naviguje pomocí horizontálního scrollu nebo pomocí kliknutí na konkrétní sekci.

Z úsporných důvodů byl použit AppBarLayout, který obsahuje Toolbar a TabLayout. Během vertikálního scrollu je Toolbar skryt a je viditelný pouze TabLayout informující uživatele o tom, v které sekci se právě nachází.

Data je potřeba průběžně obnovovat. Byl použit Timer, který v pravidelném intervalu provádí odeslání požadavku na aktuální data. Interval lze nastavit pomocí Spinneru, který je umístěn v Toolbaru místo titulku.



Obrázek 8-Zobrazení aktuálních dat zařízení

Každá stránka ViewPageru je samostatný Fragment se svým layoutem a ViewModelem. Timer je implementován v hlavní Aktivitě, aby nemusel běžet v každém Fragmentu zvlášť. Po přijetí odpovědi si Aktivita vyhledá aktivní Fragmenty a předá jim aktuální data. K zobrazení obsahu ViewPageru byl navržen adaptér.

Aktuální data ve Fragmentech se dělí do tabulek podle typu. Každá tabulka je implementována pomocí TableLayoutu a je umístěná ve vlastní kartě. Karty jsou řazeny pod sebe v LinearLayoutu, který je obalen v NestedScrollView, díky němu je zajištěno správné chování AppBarLayoutu. Jednotlivým buňkám jsou v XML layoutu napevno přiděleny proměnné. Každá hodnota je formátována pomocí ValueFormatteru, který implementuje metody umožňující správné zobrazení dat. K zobrazení hodnot byl použit TextView. Tabulky obsahují velké množství těchto komponent, pro minimalizaci atributů

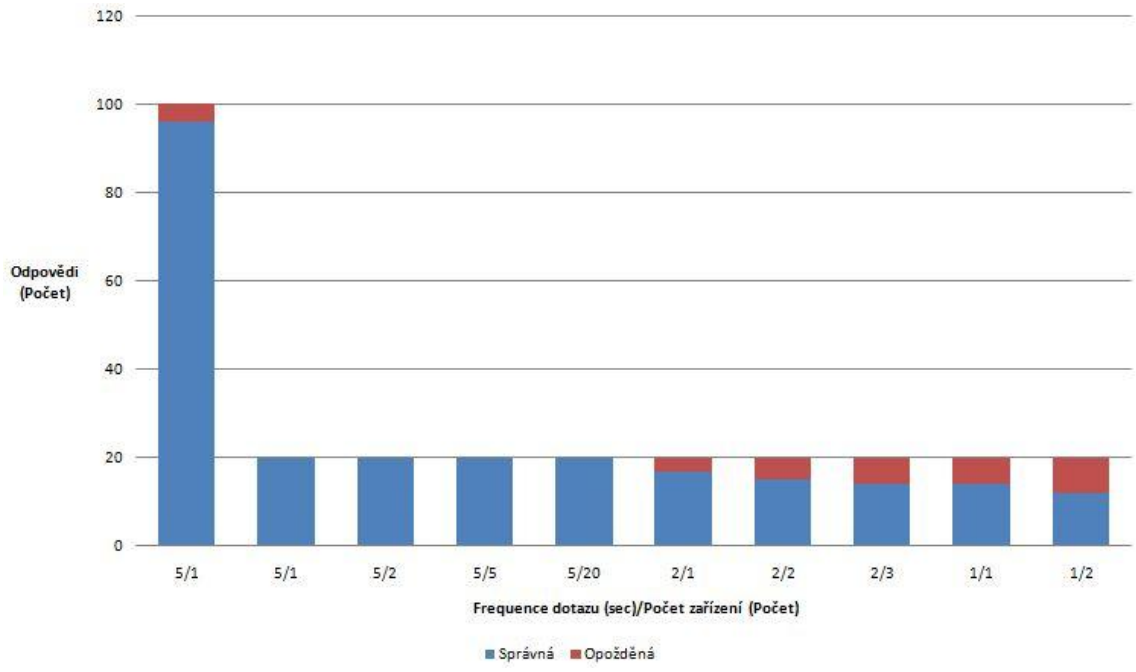
v XML layoutu byly navrženy styly. Styl pro buňku v hlavičce a styl pro buňku s daty, díky tomu má každé TextView pouze dva atributy, jeden udávající text a druhý styl.

5 Realizované experimenty, testování aplikace a výsledky

Aplikace byla podrobena sérií testů k ověření její funkčnosti a nalezení nedostatků. Byla testována chybovost při dotazování na aktuální data a vytváření CEA archivů. Během testů byl používán telefon značky Samsung, standardní emulátor mobilních telefonů z Android Studia a webový prohlížeč Google Chrome. Testovaný analyzátor je dostupný na adrese smc524.mti.tul.cz.

5.1 Aktuální data

Aktuálně měřené hodnoty jsou zapisovány do tabulek, které lze zobrazit přes detail zařízení. Jsou pravidelně obnovovány podle frekvence, která je definována Spinnerem umístěným v Toolbaru.



Obrázek 9-Naměřené hodnoty aktuálních dat

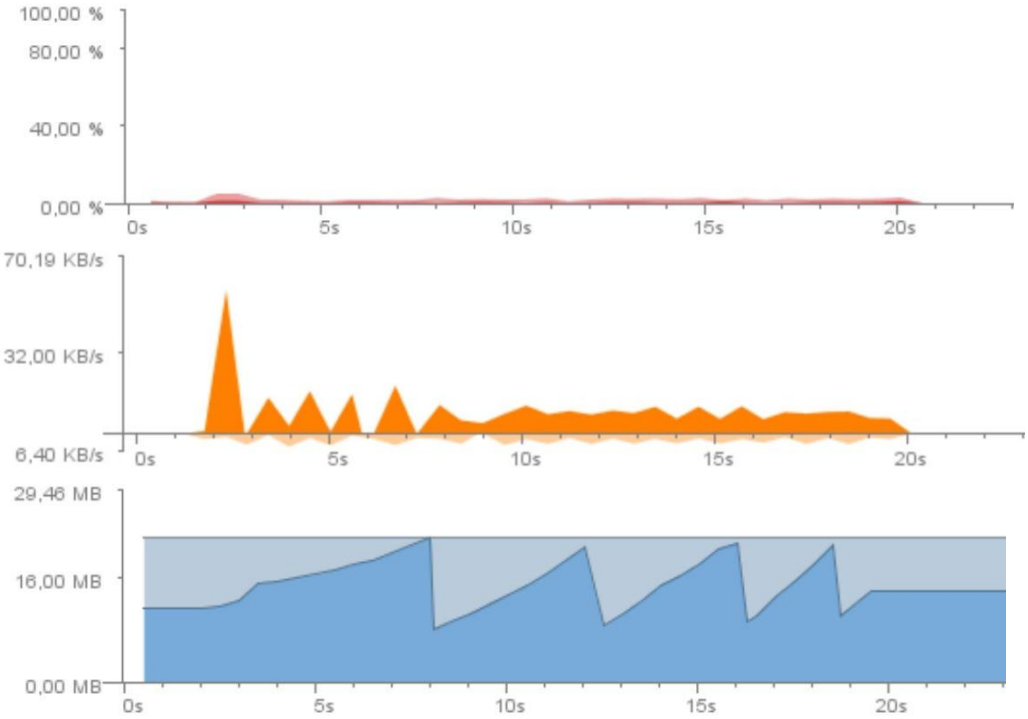
Během testů bylo sledováno zpoždění odezvy v závislosti na frekvenci obnovy a množství připojujících se zařízení. Bylo použito jedno fyzické zařízení, mobilní telefon Samsung Galaxy S6. Dále dvě virtuální zařízení generovaná pomocí emulátoru a anonymní okna prohlížeče. Použití prohlížeče

nemělo smysl při vyšší frekvenci vzhledem k nemožnosti nastavit si rychlost obnovy. Mobilní telefon využíval k datovým přenosům síť typu 4G, rychlost stahování činila 34.5 Mbps, odesílání 9.19 Mbps a ping 26.58 ms. Analyzátor SMC 144 disponuje Ethernet rozhraním s rychlostí 10 MBit, proto byla rychlost 4G sítě dostačující.

Z naměřených hodnot lze vyčíst bezproblémový chod během nižší obnovovací frekvence i během připojení více klientů. K zařízení byl připojen mobilní telefon a devatenáct oken prohlížeče, i přesto komunikace fungovala bez problémů. Změna nastala při vyšší obnovovací frekvenci. Zvyšováním frekvence a počtu připojených zařízení docházelo ke zpoždění odezvy. Dotazy byly prováděny asynchronně, což vedlo k míšení odpovědí a zobrazení neaktuálních dat. Zde se nabízí otázka, zdali je vhodné jednotlivé dotazy odesílat nezávisle nebo využít zřetězení.

5.2 Stažení CEA archivu

Stažení CEA archivu bylo prováděno na mobilním telefonu Samsung Galaxy S6. Test spočíval ve stažení dvou set záznamů hlavního archivu. Vyhodnocovalo se paměťové zatížení, využití procesoru a vytížení sítě. Stahování záznamů bylo nastaveno po jednom (4377 bajtů), komprimovaný CEA soubor měl velikost 363 kB.



Obrázek 10-Zatížení telefonu

Stahování probíhalo přibližně dvacet sekund. Během této doby byl procesor vytížen minimálně a z výsledku lze předvídat plynulý chod aplikace i na méně výkonných zařízeních. Největší zatížení procesoru nastalo při zahájení stahování, ve chvíli, kdy se vytváří struktura CEA souboru, stahují se konfigurace a zapisují potřebné soubory a XML. S tím je spojeno i zvýšené zatížení sítě. Z grafu popisujícího datový tok lze sledovat pravidelnost při odesílání požadavku a přijímání odpovědi. Posledním grafem je využití paměti. Je v něm znázorněna práce garbage collectoru během uvolňování paměti pro další stahované záznamy.

6 Závěr

V úvodu této práce byl stanoven cíl v podobě vytvoření mobilní aplikace pro komunikaci, záznam a zobrazení hodnot měření analyzátorů platformy F4 v2.0. Tato aplikace měla zjednodušit práci se zařízením, které samo o sobě nemá velké zobrazovací schopnosti.

Stanovený cíl byl splněn a výsledkem je mobilní aplikace pro platformu Android. Aplikace implementuje lokální databázi umožňující uživateli správu svých zařízení. Byla navržena komunikační knihovna v jazyce Java, která umožňuje stažení a uložení záznamů dat, stažení aktuálních dat, konfigurací a informací o analyzátoru. Tato knihovna byla implementována v mobilní aplikaci, díky ní je uživateli umožněno se připojit a stáhnout data, která si zvolí, nebo sledovat aktuální dění v analyzátoru. Aby vše bylo konzistentní, byla využita support knihovna od Googlu obsahující základní stavební prvky pro Android aplikace.

V práci tak byly splněny všechny body zadání a výsledná aplikace byla podrobena testům k ověření spolehlivosti a zátěže zařízení během stahování a vytváření CEA archivu.

Aplikaci je možné dále rozšiřovat. Lze zlepšit její zobrazovací schopnosti aktuálních dat nebo se zamyslet nad zobrazováním stažených archivů. Zde by mohl nastat problém v podobě omezené paměti mobilního zařízení, protože operační systém Android přiděluje jednotlivým aplikacím poměrně malou kapacitu RAM. Rozšiřovat je možné i lokální databázi analyzátorů a umožnit tak uživateli například řadit si zařízení do skupin.

Seznam použité literatury

1. Facility Insights services. *Schneider Electric Global Website.* [Online] Schneider Electric. [Citace: 5. květen 2016.] http://www.schneider-electric.com/en/product-range/63092-facility-insights-services/?parent-category-id=4100.

2. Residential. *wattwatchers digital energy.* [Online] Wattwatchers. [Citace: 4. květen 2016.] http://wattwatchers.com.au/residential/.

3. *Curb.* [Online] Curb. [Citace: 4. květen 2016.] http://energycurb.com/.

4. Dranetz HDPQ® Visa Power Quality Analyzer. *Dranetz.* [Online] Dranetz. [Citace: 4. květen 2016.] http://www.dranetz.com/product-services/current-dranetz-products/dranetz-hdpq-visa/.

5. Power Quality Analyzer. *Kyoritsu.* [Online] Kyoritsu. [Citace: 5. květen 2016.] http://www.kew-ltd.co.jp/en/products/powermeter/6315.html.

6. Power Quality Analyzers. *ht-instruments.* [Online] HT. [Citace: 5. květen 2016.] http://www.ht-instruments.com/en/products/power-quality-analyzers/wifi/pqa819/.

7. PSM-i PowerSight iPad Software App. *PowerSight.* [Online] PowerSight. [Citace: 5. květen 2016.] http://www.powersight.com/HTML/PRODUCTS/PSM-i%20power%20monitor%20software%20for%20iPad.html.

8. SMV, SMP, SMPQ Multifunctional Panel Meters & Power Quality Analyzers Communication Protocol Manual. *KMB systems.* [Online] 29. červenec 2009. [Citace: 10. leden 2016.] http://www.kmb.cz/index.php/en/download-en/category/12-multifunction-panel-meters.

9. CoordinatorLayout. *Android Developers.* [Online] Google Inc. [Citace: 18. duben 2016.] http://developer.android.com/reference/android/support/design/widget/CoordinatorLayout.html.

10. **Lake, Ian.** Intercepting everything with CoordinatorLayout Behaviors. *Medium.* [Online] 17. únor 2016. [Citace: 18. únor 2016.] https://medium.com/google-developers/intercepting-everything-with-coordinatorlayout-behaviors-8c6adc140c26#.6shdjby7j.

11. AppBarLayout. *Android Developers.* [Online] Google Inc. [Citace: 18. duben 2016.] http://developer.android.com/reference/android/support/design/widget/AppBarLayout.html.

12. CollapsingToolbarLayout. *Android Developers.* [Online] Google Inc. [Citace: 18. duben 2016.] http://developer.android.com/reference/android/support/design/widget/CollapsingToolbarLayout.html.

13. Toolbar. *Android Developers.* [Online] Google Inc. [Citace: 18. duben 2016.] http://developer.android.com/reference/android/widget/Toolbar.html.

14. RecyclerView. *Android Developers.* [Online] Google Inc. [Citace: 18. duben 2016.] https://developer.android.com/reference/android/support/v7/widget/RecyclerView.html.

15. CardView. *Android Developers.* [Online] Google Inc. [Citace: 18. duben 2016.] http://developer.android.com/reference/android/support/v7/widget/CardView.html.

16. FloatingActionButton. *Android Developers.* [Online] Google Inc. [Citace: 18. duben 2016.] http://developer.android.com/reference/android/support/design/widget/FloatingActionButton.html.

17. Návrhové vzory. *Algoritmy.net.* [Online] INFO WEB s.r.o. [Citace: 3. květen 2016.] https://www.algoritmy.net/article/51224/Navrhove-vzory.

18. **Jurisic, Josip.** Architecting Android with Data Binding and MVVM in mind. *Medium.* [Online] 18. únor 2016. [Citace: 19. únor 2016.] https://medium.com/cobe-mobile/architecting-android-with-data-binding-and-mvvm-in-mind-8874bbec0b0d#.chusrhyr9.

19. **Leiva, Antonio.** MVP for Android: how to organize the presentation layer. *Antonio Leiva.* [Online] 15. duben 2014. [Citace: 18. duben 2016.] http://antonioleiva.com/mvp-android/.

20. **Kinšt, Jakub.** Android-ViewModelBinding. *GitHub.* [Online] 19. leden 2016. [Citace: 12. únor 2016.] https://github.com/jakubkinst/Android-ViewModelBinding.

Seznam příloh

- Příloha A – CD-ROM
- Příloha B – Instalace a distribuce aplikace
- Příloha C – Ukázka layoutu ve více rozlišeních

Příloha A – CD-ROM

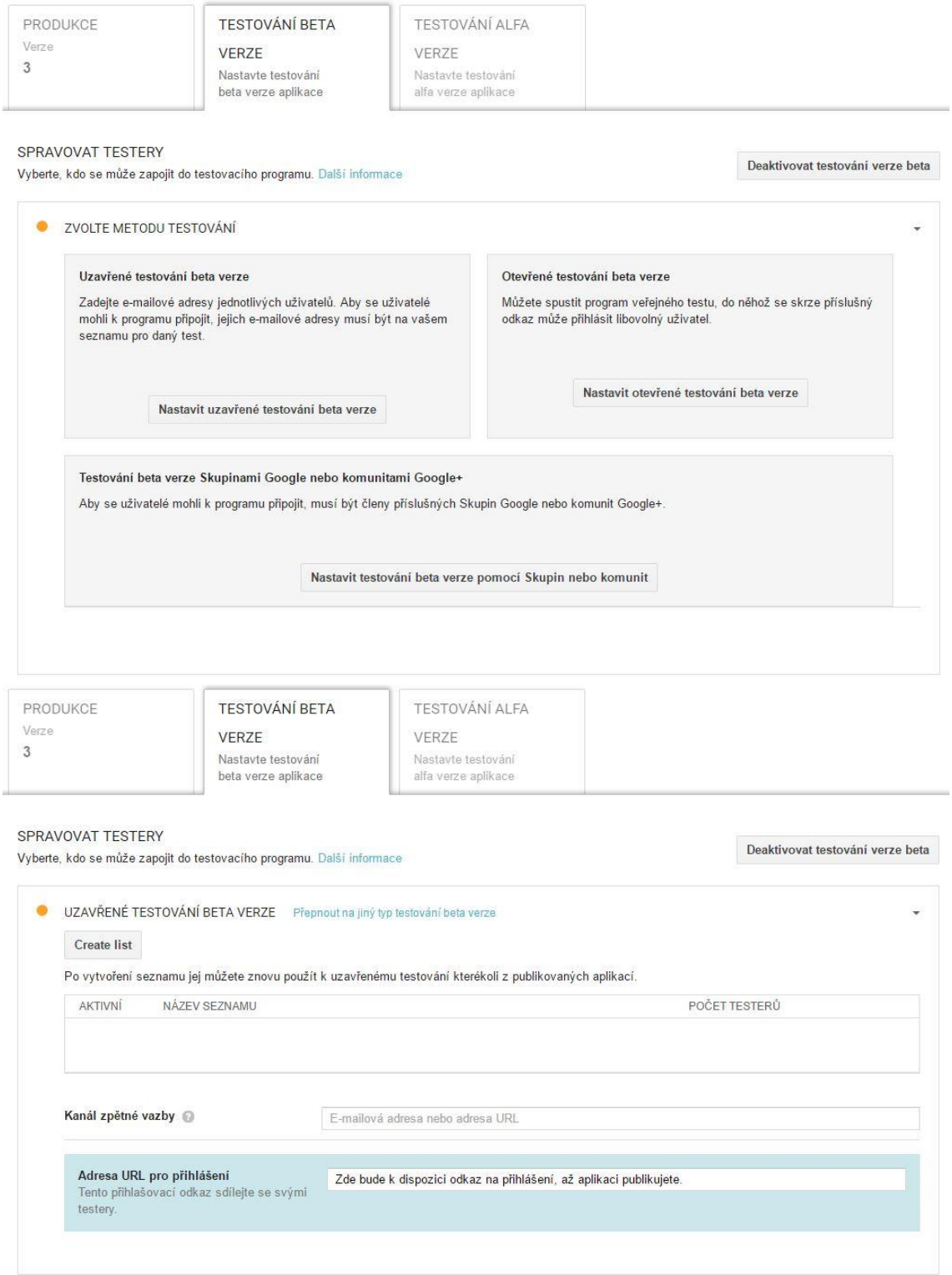
Obsahem přiloženého CD-ROM jsou následující složky:

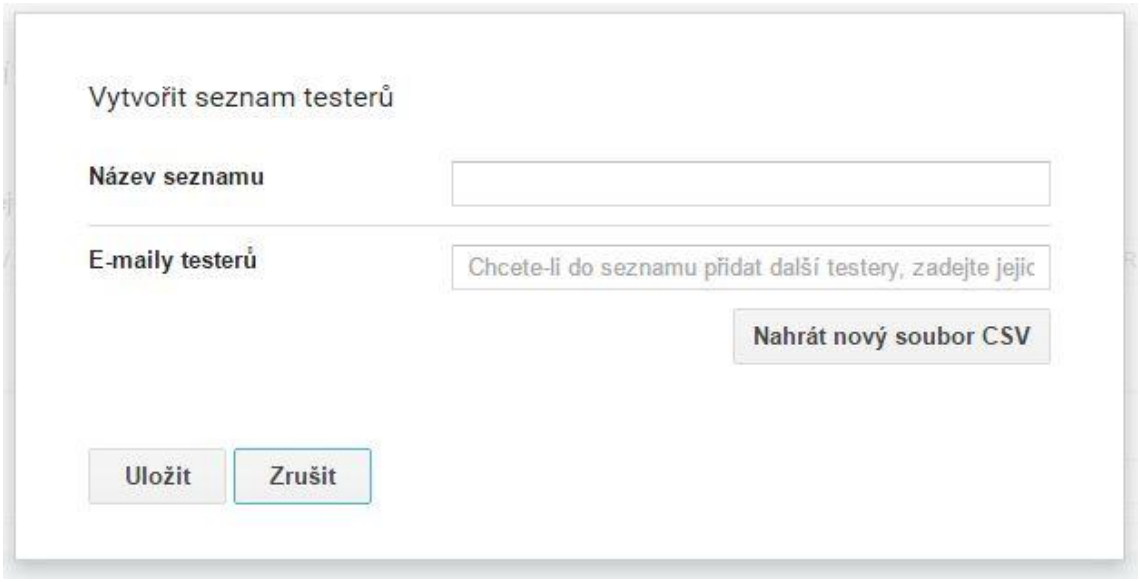
- diplomová práce – obsahuje PDF verzi diplomové práce

Příloha B – Instalace a distribuce aplikace

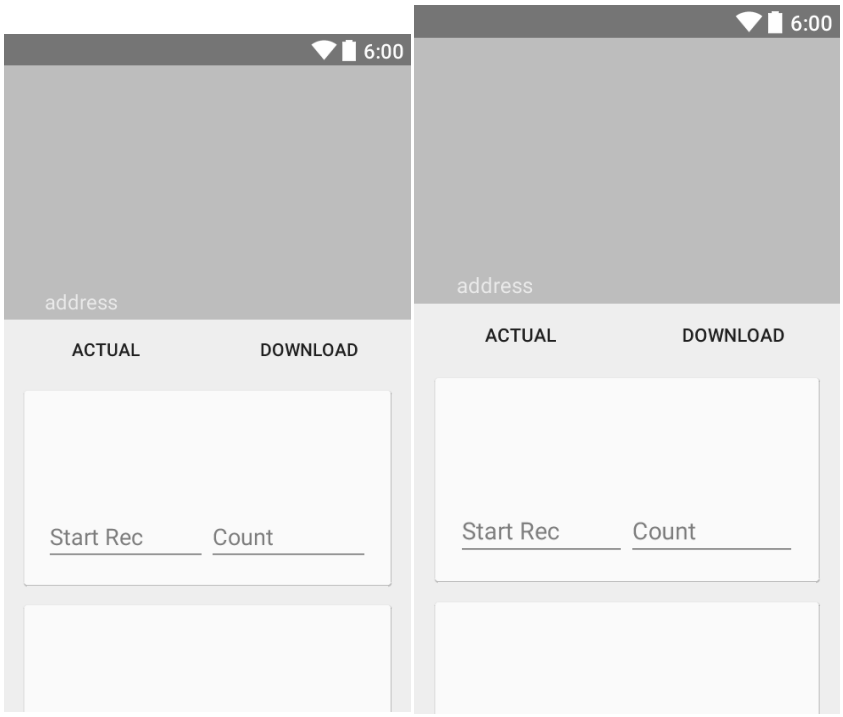
Aplikaci lze distribuovat pomocí Google Play. Je nutné vyplnit základní informace jako jsou název, typ, hodnocení obsahu z hlediska závadnosti, nahrání několika obrázků aplikace, nahrání loga, nahrání banneru atd. Samotný instalační soubor musí být podepsaný pomocí vývojářského certifikátu a aplikace musí mít unikátní package name. Aby byla aplikace viditelná pouze pro vybrané uživatele je nutné publikovat ji v beta testu. Uživatele, kteří budou mít k aplikaci přístup, jsou určení jejich Google účtem.

Aplikace vyžaduje Android verze 4.2 a vyšší, připojení k internetu, povolení pro čtení a zápis do paměti telefonu a povolení k zakázání uspání mobilního telefonu.

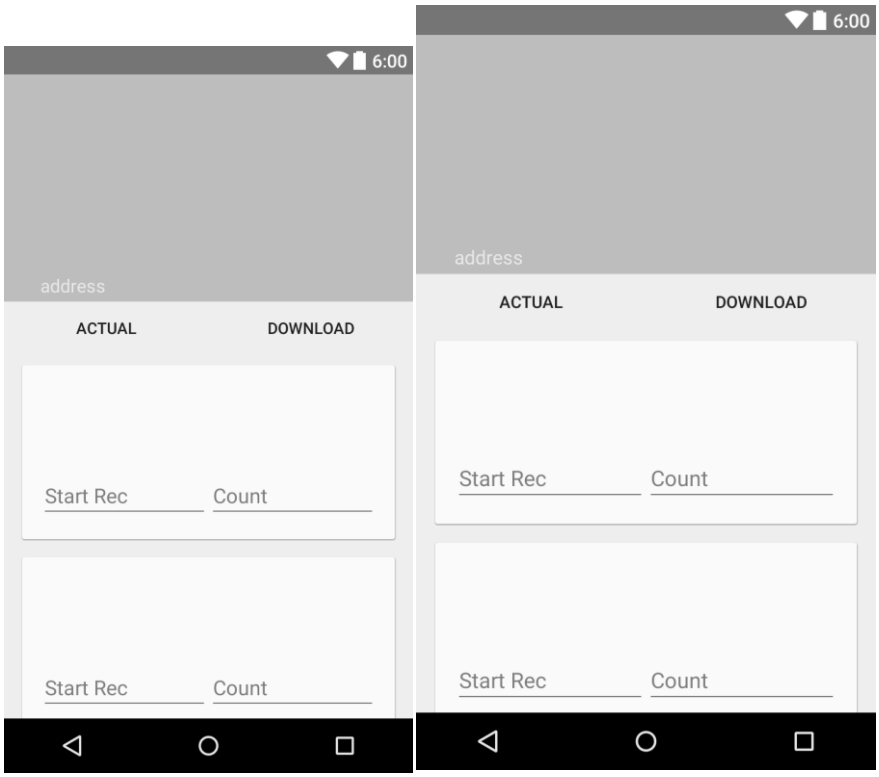

PRODUKCE
Verze
3
TESTOVÁNÍ BETA
VERZE
Nastavte testování beta verze aplikace
TESTOVÁNÍ ALFA
VERZE
Nastavte testování alfa verze aplikace
SPRAVOVAT TESTERY
Vyberte, kdo se může zapojit do testovacího programu. Další informace
Deaktivovat testování verze beta
ZVOLTE METODU TESTOVÁNÍ
Uzavřené testování beta verze
Zadejte e-mailové adresy jednotlivých uživatelů. Aby se uživatelé mohli k programu připojit, jejich e-mailové adresy musí být na vašem seznamu pro daný test.
Nastavit uzavřené testování beta verze
Otevřené testování beta verze
Můžete spustit program veřejného testu, do něhož se skrze příslušný odkaz může přihlásit libovolný uživatel.
Nastavit otevřené testování beta verze
Testování beta verze Skupinami Google nebo komunitami Google+
Aby se uživatelé mohli k programu připojit, musí být členy příslušných Skupin Google nebo komunit Google+.
Nastavit testování beta verze pomocí Skupin nebo komunit
PRODUKCE
Verze
3
TESTOVÁNÍ BETA
VERZE
Nastavte testování beta verze aplikace
TESTOVÁNÍ ALFA
VERZE
Nastavte testování alfa verze aplikace
SPRAVOVAT TESTERY
Vyberte, kdo se může zapojit do testovacího programu. Další informace
Deaktivovat testování verze beta
UZAVŘENÉ TESTOVÁNÍ BETA VERZE
Přepnout na jiný typ testování beta verze
Create list
Po vytvoření seznamu jej můžete znovu použít k uzavřenému testování kterékoli z publikovaných aplikací.
AKTIVNÍ
NÁZEV SEZNAMU
POČET TESTERŮ
Kanál zpětné vazby
E-mailová adresa nebo adresa URL
Adresa URL pro přihlášení
Tento přihlašovací odkaz sdílejte se svými testery.
Zde bude k dispozici odkaz na přihlášení, až aplikaci publikujete.


Vytvořit seznam testerů
Název seznamu
E-maily testerů
Chcete-li do seznamu přidat další testery, zadejte jejic
Nahrát nový soubor CSV
Uložit
Zrušit

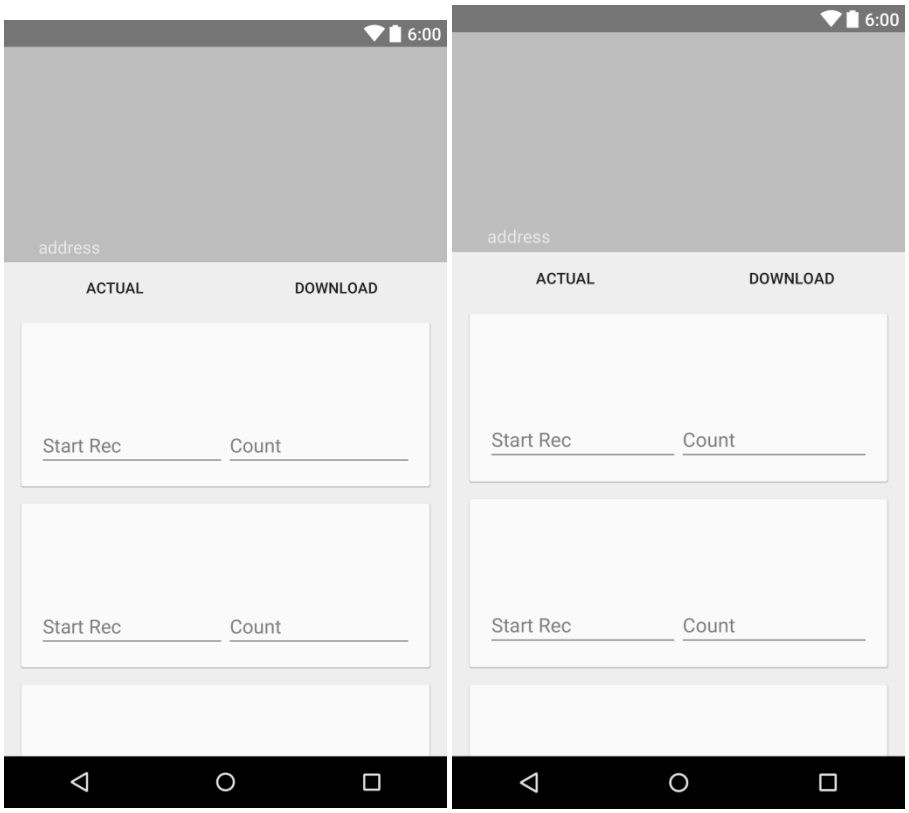
Příloha C – Ukázka layoutu ve více rozlišeních



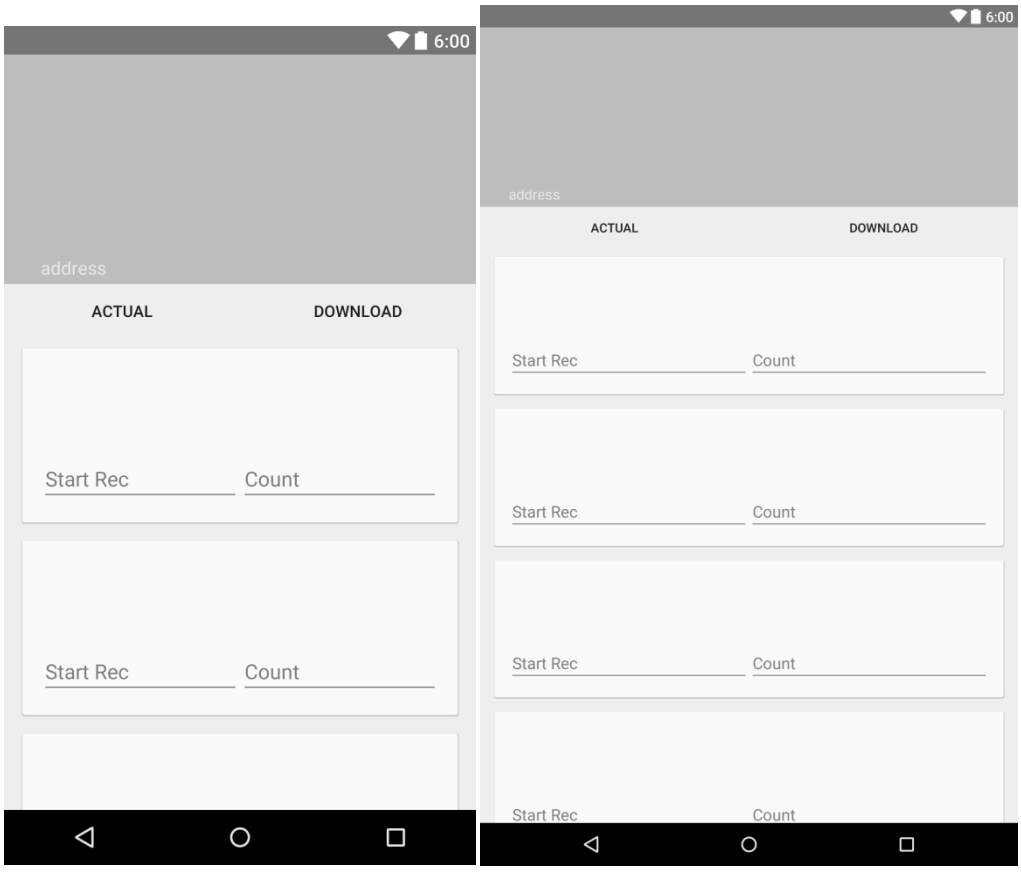
Ukázka layoutu 1-Nexus One 3,7", Nexus S 4“



Ukázka layoutu 2-Galaxy Nexus 4,7", Nexus 5 5"



Ukázka layoutu 3-Nexus 5X 5,2", Nexus 6 6"



Ukázka layoutu 4-Nexus 6P 5,2", Nexus 7 7"